Machine Learning: Foundations, Methodologies, and Applications

Teik Toe Teoh
Yu Jin Goh

# Artificial Intelligence in Business Management

Springer

# Machine Learning: Foundations, Methodologies, and Applications

Books published in this series focus on the theory and computational foundations, advanced methodologies and practical applications of machine learning, ideally combining mathematically rigorous treatments of a contemporary topics in machine learning with specific illustrations in relevant algorithm designs and demonstrations in real-world applications. The intended readership includes research students and researchers in computer science, computer engineering, electrical engineering, data science, and related areas seeking a convenient medium to track the progresses made in the foundations, methodologies, and applications of machine learning.

Topics considered include all areas of machine learning, including but not limited to:

- Decision tree
- Artificial neural networks
- Kernel learning
- Bayesian learning
- Ensemble methods
- Dimension reduction and metric learning
- Reinforcement learning
- Meta learning and learning to learn
- Imitation learning
- Computational learning theory
- Probabilistic graphical models
- Transfer learning
- Multi-view and multi-task learning
- Graph neural networks
- Generative adversarial networks
- Federated learning

This series includes monographs, introductory and advanced textbooks, and state-of-the-art collections. Furthermore, it supports Open Access publication mode.

Teik Toe Teoh • Yu Jin Goh

# Artificial Intelligence in Business Management

Springer

Teik Toe Teoh
Information Technology & Operations Management
Nanyang Technological University
Singapore, Singapore

Yu Jin Goh
Singapore, Singapore

ISSN 2730-9908 ISSN 2730-9916 (electronic)
Machine Learning: Foundations, Methodologies, and Applications
ISBN 978-981-99-4557-3 ISBN 978-981-99-4558-0 (eBook)
https://doi.org/10.1007/978-981-99-4558-0



This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Paper in this product is recyclable.

# Preface

This textbook is a practical guide for Artificial Intelligence in business management, co-authored by Dr Teoh Teik Toe from Nanyang Technological University and Goh Yu Jin. It provides many learning materials and practical examples accumulated over the years in applying Artificial Intelligence to business applications as practitioners.

Dr Teoh has been conducting research in Big Data, Deep Learning, Cybersecurity, Artificial Intelligence, Machine Learning, and Software Development for more than 25 years. His works have been published in more than 50 journals, conference proceedings, books, and book chapters. He holds a PhD in Computer Engineering from Nanyang Technological University (NTU); a Doctor of Business Administration from the University of Newcastle; a Master of Law from the National University of Singapore (NUS); LLB and LLM from the University of London (UoL); and Chartered Financial Analyst (CFA), Association of Chartered Certified Accountants (ACCA), and Chartered Institute of Management Accountants (CIMA) qualification. With over 15 years of experience in Data Mining, Quantitative Analysis, Data Statistics, Finance, Accounting, and Law, he is passionate about the synergy between business and technology. He believes that Artificial Intelligence should be accessible to everyone and is eager to share his knowledge of the field.

Throughout his career, Dr Teoh has taught Artificial Intelligence to students from all backgrounds and levels of experience, including those with limited programming knowledge. Under his guidance, many of his students gained confidence in writing their own AI programs.

Yu Jin is an Artificial Intelligence Engineer with four years of working experience. He enjoys the challenges of using data to improve business processes and has 1.5 years of teaching experience in AI and Data Science. He coauthored two conference publications on AI and deep learning and is passionate about applying AI to a variety of use cases and making it accessible to everyone.

Artificial Intelligence is a broad field that involves creating systems capable of executing tasks that typically require human intelligence. With applications in time series, images, videos, speech, and text data, AI has become versatile enough to be applied across all types of businesses. However, developing develop good Artificial

Intelligence programs requires a deep understanding of the field. While concepts used in self-driving cars and virtual assistants like Amazon's Alexa may seem very complex and difficult to grasp, this book will cover them in a step-by-step manner with code snippets to provide tangible examples for readers to learn from. This approach ensures that the lessons are easy to follow.

This book aims to provide readers with an easy-to-understand explanation on how Artificial Intelligence can be applied to alleviate various pain points faced by businesses. We hope to empower individuals who are eager to learn about Artificial Intelligence and apply it to solve business problems. By introducing readers to the potential of Artificial Intelligence, this book seeks to help them understand how to make the most of the technology and how it can be used for their business. Through the materials compiled in this book, readers will gain the knowledge they need on how to create such systems to solve business problems.

Part I of the book introduces readers to various key concepts required to understand how Artificial Intelligence algorithms are developed. Topics in Artificial Intelligence such as classification, regression, and clustering are covered to build a solid foundation for beginners. Readers will be taught various methods of applying Artificial Intelligence to unstructured data to handle a wide variety of data appropriately.

This builds the foundation for Part II, where readers will be exposed to how Artificial Intelligence can be applied to various aspects of business. Various implementations of Artificial Intelligence are shared, allowing readers to generate their own Artificial Intelligence algorithms to perform various tasks such as lead scoring, contract analysis, SWOT analysis using reviews, and product demand forecasting.

Singapore, Singapore

Teik Toe Teoh
Yu Jin Goh

# Acknowledgments

We would like to express our gratitude to the many individuals who have enabled and supported us throughout the process of writing and publishing this work.

Firstly, we would like to thank all the people who have helped to collect and prepare the data used in the examples demonstrated within this book. The step-by-step examples written in this book can only be created because of your contributions in developing these datasets for people to practice Artificial Intelligence with.

Secondly, our tutorials and code were adapted from various sources. We would thus like to thank all the developers that we have learned from in the process of writing this book. Without the work of the authors in the references, our book would not have been possible.

Thirdly, we would like to extend our appreciation to the Springer team for the invaluable support and guidance provided in turning this book idea into reality.

Fourthly, we would like to thank our colleagues and bosses who have demonstrated their support in favor of writing this book. This book would not have been realized without the support we have received and we are deeply grateful for the opportunity.

Lastly, we would like to thank all our families and friends for their support and encouragement throughout this endeavor. Their love, support, and encouragement has helped motivate us to finish writing this book. Goh Yu Jin would like to thank his family, Goh Huang Sheng, Loh Lin See, and Goh Yan Xian, for the support they have provided during this journey.

# Contents

# Part I
# Artificial Intelligence Algorithms

# Chapter 1
# Introduction to Artificial Intelligence


## 1.1 Introduction

Intelligence refers to the capability of acquiring knowledge and applying them to solve difficult challenges. upGrad [1] Humans are highly intelligent beings that have been learning about and developing solutions to various problems for many centuries. Artificial intelligence is humanity's attempt to replicate this powerful innate ability to learn in our machines.

Machines that can learn by themselves would be able to deepen their knowledge and make better decisions. Furthermore, combining artificial intelligence with robotics would allow us to benefit from the increased efficiency and automation of manual labor that it brings.

Through the use of artificial intelligence, it would be possible to develop solutions that are able to function autonomously without requiring human intervention.

The field of artificial intelligence is very broad and covers a multitude of topics. There are many different types of basic intelligence that is present in humans that artificial intelligence engineers hope to be able to recreate in their creations such as:

- Visual-spatial intelligence
- Linguistic intelligence
- Logical intelligence
- Bodily kinesthetic intelligence
- Interpersonal intelligence

Developing artificial intelligence solutions requires a high amount of upfront investment to prepare and clean the data. This process is shown in Fig. 1.1. In order to develop a useful model, the data collected much first be prepared in a way that these algorithms can use them, and feature extraction has to be performed to sieve out useful features, followed by model training and eventually model testing.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_1

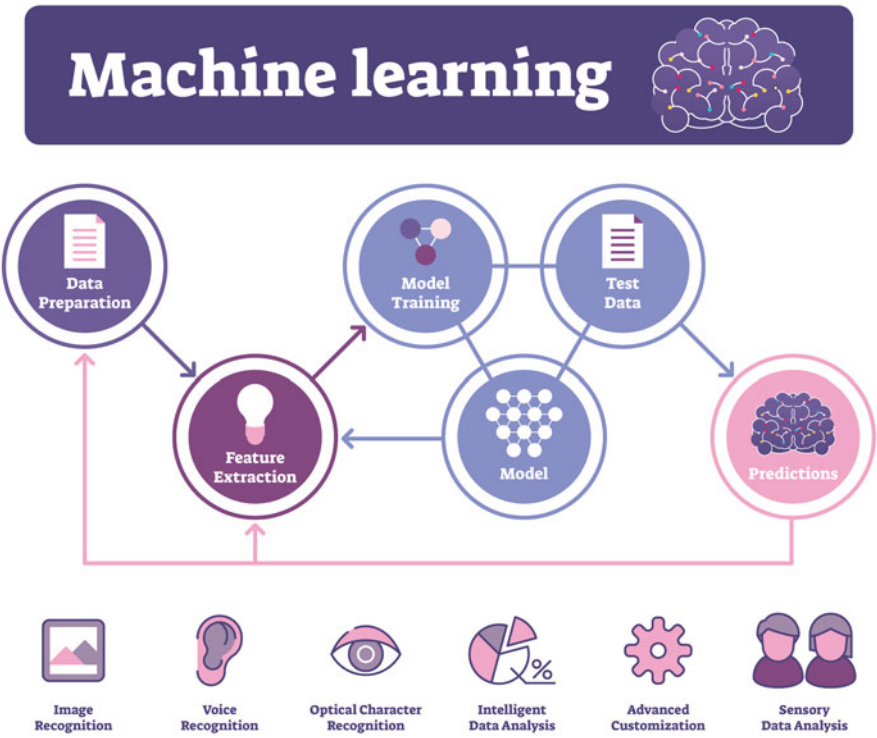



**Fig. 1.1** Process and applications of machine learning [2]

These algorithms can be developed for a wide variety of tasks across different data types as well. Some examples of tasks that can be used by artificial intelligence algorithms are:

1. Image recognition
2. Voice recognition
3. Optical character recognition
4. Intelligent data analysis
5. Customized data types
6. Sensory data analysis

Artificial intelligence algorithms are commonly applied to solve one of 4 different types of problems shown in Fig. 1.2:

1. **Classification**
   Classification tasks are used in categorizing the input data. The algorithms aim to categorize the data into various pre-defined categories that could be binary (e.g., spam vs. not spam) or multi-class such as (e.g., dog, cat, bird, fish).

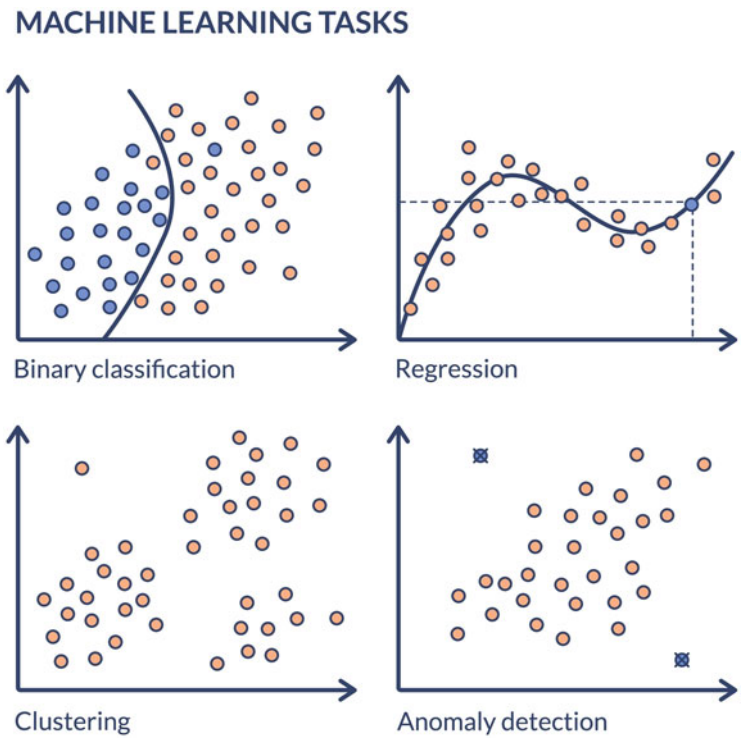



**Fig. 1.2** Types of machine learning tasks [3]

2. **Regression**
   Regression tasks are used to predict real or continuous values such as age and salary. Further analysis can be done to understand how the dependent variable will be impacted by the independent variable.
3. **Clustering**
   Clustering aims to group data points into certain groups based on how similar they are from one another. It is generally an unsupervised learning problem and aims to identify naturally occurring patterns in the data.
4. **Anomaly detection**
   Anomaly detection aims to identify anomalies by learning what is normal and predicting all things that fall out of normality as abnormal. Anomalies may not occur very often, and so specialized techniques are developed to try to solve these tasks.

## 1.2 History of Artificial Intelligence

One of the challenges in developing intelligent solutions is being able to measure its success. In 1950, Alan Turing developed the Turing Test that helped to determine whether or not a machine is intelligent.

Many other scholars and mathematicians apart from Turing were deeply interested in this topic. The phrase "artificial intelligence" was created during a symposium in 1956. For many years, governments incubated the growth of artificial intelligence solutions up till the 1974 when the first AI winter began. AI research ceased at the time due to apathy and sluggish development of the field. However, interest in artificial intelligence picked up soon after using if-then logic that caused a boom in 1980 and faded in 1988 once the limitations of if-then logic surfaced. Artificial intelligence began to pick up greatly in 1995 when computational power and big data became widely available, thus increasing the possibilities for more complex artificial intelligence solutions to be developed.

AI has grown by leaps and bounds since then. In 1997, an IBM computer called Deep Blue defeated Russian champion Garry Kasparov at the game of chess. In 2016, Google's AlphaGo had beaten a World Champion, Lee Sedol, at the game of Go.

Nowadays, artificial intelligence can be found everywhere from mobile phones to healthcare devices in hospitals. Numerous businesses are focusing on leveraging artificial intelligence to build better solutions and tackle various difficulties they face.

## 1.3 Types of Artificial Intelligence Algorithms

There are generally 3 main types of artificial intelligence algorithms as shown in Fig. 1.3. Each of them comes with its own benefits and drawbacks:

1. **Supervised learning**
   Supervised learning algorithms are the most widely used type of algorithms among the three. These algorithms are made to learn from human labeled data in an attempt to recognize patterns in the data required to predict and mimic the answers provided by the labelers. These algorithms are highly effective in learning the task at hand but require high amounts of labor from annotators in labeling the huge amounts of data required.
2. **Unsupervised learning**
   Unsupervised learning algorithms are designed to learn from data without labels to identify naturally occurring patterns and clusters in the underlying data. Such algorithms can be highly effective in uncovering naturally occurring patterns from the data and can easily use large amounts of data as do not require the huge annotation effort required in supervised learning. However, it might be difficult to understand the results presented, and it may be hard to adapt to a specific task

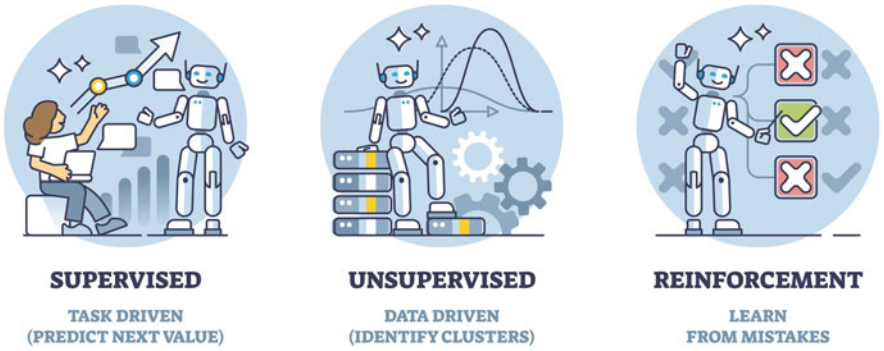



**Fig. 1.3** Types of artificial Intelligence Algorithms [4]

as the algorithm does not know what we want it to learn from the underlying data.

3. **Reinforcement learning**
   Reinforcement learning algorithms are developed using a reward and punishment system. These algorithms learn by maximizing the rewards it can achieve when performing certain actions and minimizing the penalties in performing actions where punishments are dealt. Such algorithms are effective in learning tasks that can be hard to define which humans may not know how to go about achieving it. On the other hand, it is less effective in learning tasks which answers are provided for as compared to supervised algorithms since the algorithm also has to test and explore new outcomes. Deep Blue and AlphaGo are examples of reinforcement learning algorithms. ChatGPT was also finetuned using reinforcement learning to further improve user experience.

## 1.4 Organization of the Book

Artificial intelligence in business management aims to help management professionals exploit the predictive powers of AI and demonstrate to AI practitioners how their expertise can be applied in fundamental business operations. As such, the book has been organized into two parts: artificial intelligence algorithms and applications of artificial intelligence in business management.

Part I of the book aims to help readers build up their knowledge of common artificial intelligence algorithms, through the explanations of various algorithms and simple python examples provided. On the other hand, Part II highlights different problems faced by business management professionals across core business functions and illustrates how artificial intelligence could be utilized to alleviate them.

The content covered in Part I would be essential to business management professionals, which may not be as familiar with artificial intelligence, to gain

a better understanding of how different kinds of artificial intelligence algorithms work. Readers would be exposed to different problems and types of data where artificial intelligence algorithms are used to solve. This knowledge acquired in Part I will be especially valuable in Part II, as these algorithms will be utilized to help solve various specific challenges in different business contexts. Therefore, Part II will be focused on how AI can be utilized to alleviate different kinds of issues faced by businesses.

In this way, basic concepts and simple problems in artificial intelligence will be covered first before working on applying AI to more complex business problems to facilitate learning. We hope that by reading this book, readers will be able to better leverage the potential of AI for business success.

## References

1. upGrad (2020) Introduction to AI: History, components, pros & cons, examples & applications. In: Medium. https://medium.com/@upGrad/introduction-to-ai-history-components-pros-cons-examples-applications-c626692fcc86. Accessed 27 Apr 2023
2. VectorMine (2023) Free machine learning vector illustration – VectorMine. https://vectormine.com/item/machine-learning-illustrated-vector-diagram-with-icons/
3. VectorMine (2023) Machine Learning Tasks Graphic Scheme Classic Stock Vector (Royalty Free) 2215598449 | Shutterstock. https://www.shutterstock.com/image-vector/machine-learning-tasks-graphic-scheme-classic-2215598449. Accessed 13 May 2023
4. VectorMine (2023) Types Machine Learning Algorithms Classification Outline Stock Vector (Royalty Free) 2220216845 | Shutterstock. https://www.shutterstock.com/image-vector/types-machine-learning-algorithms-classification-outline-2220216845. Accessed 13 May 2023

# Chapter 2
# Regression


**Learning Outcomes**

- Learn and apply basic models for regression tasks using sklearn and keras.
- Learn data processing techniques to achieve better results.
- Learn how to use simple feature selection techniques to improve our model.
- Data cleaning to help improve our model's RMSE .

Regression looks for relationships among variables. For example, you can observe several employees of some company and try to understand how their salaries depend on the features, such as experience, level of education, role, city they work in, and so on.

This is a regression problem where data related to each employee represent one observation. The presumption is that the experience, education, role, and city are the independent features, and the salary of the employee depends on them.

Similarly, you can try to establish a mathematical dependence of the prices of houses on their areas, the numbers of bedrooms, distances to the city center, and so on.

Generally, in regression analysis, you usually consider some phenomenon of interest and have a number of observations. Each observation has two or more features. Following the assumption that (at least) one of the features depends on the others, you try to establish a relation among them.

The dependent features are called the dependent variables, outputs, or responses.

The independent features are called the independent variables, inputs, or predictors.

Regression problems usually have one continuous and unbounded dependent variable. The inputs, however, can be continuous, discrete, or even categorical data such as gender, nationality, brand, and so on.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_2

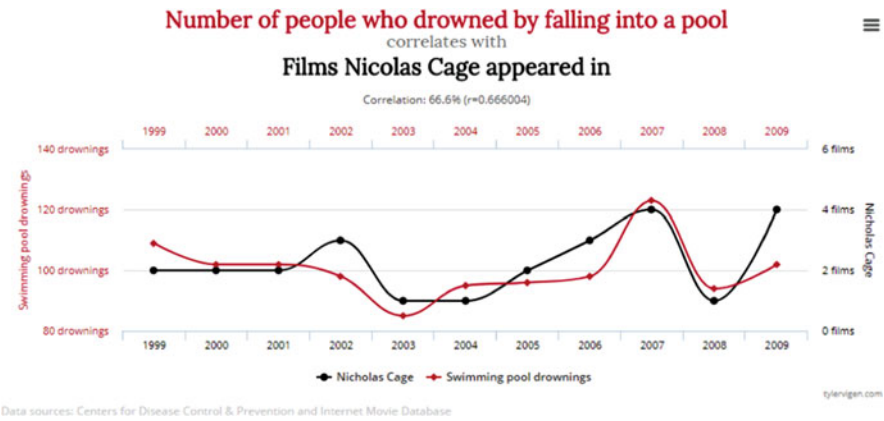



**Fig. 2.1** Example of spurious correlations [5]

It is a common practice to denote the outputs with $y$ and inputs with $x$. If there are two or more independent variables, they can be represented as the vector $\mathbf{x} = (x_1, \ldots, x_r)$, where $r$ is the number of inputs.

Typically, you need regression when you need to answer whether some phenomenon influences the other and how do they influence each other. You may also need regression to understand how several variables are related. For example, you can use it to determine if and to what extent the experience or gender impacts salaries.

Regression is also useful when you want to forecast a response using a new set of predictors. For example, you could try to predict electricity consumption of a household for the next hour given the outdoor temperature, time of day, and the number of residents in that household.

Regression is used in many different fields: economy, computer science, social sciences, and so on. Its importance rises every day with the availability of large amounts of data and increased awareness of the practical value of data.

An important to note is that regression does not imply causation. It is easy to find examples of non-related data that, after a regression calculation, do pass all sorts of statistical tests. The following Fig. 2.1 is a popular example that illustrates the concept of data-driven "causality."

It is often said that correlation does not imply causation, although, inadvertently, we sometimes make the mistake of supposing that there is a causal link between two variables that follow a certain common pattern.

**Dataset: "Alumni Giving Regression (Edited).csv"** You can obtain the dataset from this link:

```
https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni
↪%20Giving%20Regression%20%28Edited%29.csv?dl=0.
```

Also, you may run the following code in order to download the dataset in `google colab`:

```
!wget https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni
↪%20Giving%20Regression%20%28Edited%29.csv?dl=0 -O
--quiet  "./Alumni Giving Regression (Edited).csv"
```

```
!wget https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni
↪%20Giving%20Regression%20%28Edited%29.csv?dl=0 -O -
↪quiet  "./Alumni Giving Regression (Edited).csv"
```

```
# Importing libraries needed
# Note that keras is generally used for deep learning␣
↪as well
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report,␣
↪confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor,
↪GradientBoostingRegressor
import pandas as pd
import csv
```

```
Using TensorFlow backend.
```

In general, we will import dataset for structured dataset using pandas. We will also demonstrate the code for loading dataset using numpy to show the differences between both libraries. Here, we are using a method in pandas call `read_csv` that takes the path of a csv file. `'CS'` in `CSV` represents comma separated. Thus, if you open up the file in excel, you would see values separated by commas.

```
# fix random seed for reproducibility
np.random.seed(7)
df = pd.read_csv("Alumni Giving Regression (Edited).
↪csv", delimiter="," )
df.head()
```

```
    A     B     C     D     E     F
0  24  0.42  0.16  0.59  0.81  0.08
1  19  0.49  0.04  0.37  0.69  0.11
2  18  0.24  0.17  0.66  0.87  0.31
3   8  0.74  0.00  0.81  0.88  0.11
4   8  0.95  0.00  0.86  0.92  0.28
```

In pandas, it is very convenient to handle numerical data. Before doing any model, it is good to take a look at some of the dataset's statistics to get a "feel" of the data. Here, we can simply call df.describe that is a method in pandas `dataframe`.

```
df.describe()
```

```
                A           B           C           D           E           F
count  123.000000  123.000000  123.000000  123.000000  123.000000  123.000000
mean    17.772358    0.403659    0.136260    0.645203    0.841138    0.141789
std      4.517385    0.133897    0.060101    0.169794    0.083942    0.080674
min      6.000000    0.140000    0.000000    0.260000    0.580000    0.020000
25%     16.000000    0.320000    0.095000    0.505000    0.780000    0.080000
50%     18.000000    0.380000    0.130000    0.640000    0.840000    0.130000
75%     20.000000    0.460000    0.180000    0.785000    0.910000    0.170000
max     31.000000    0.950000    0.310000    0.960000    0.980000    0.410000
```

Furthermore, pandas provides a helpful method to calculate the pairwise correlation between 2 variables.

Correlation refers to a mutual relationship or association between quantities (numerical number). In almost any business, it is very helping to express one quantity in terms of its relationship with others. We are concerned with this because business plans and departments are not isolated! For example, sales might increase when the marketing department spends more on advertisements, or a customer's average purchase amount on an online site may depend on his or her characteristics. Often, correlation is the first step to understanding these relationships and subsequently building better business and statistical models.

For example: "D" and "E" have a strong correlation of 0.93 that means that when D moves in the positive direction E is likely to move in that direction too. Here, notice that the correlation of `A` and `A` is 1. Of course, `A` would be perfectly correlated with `A`.

```
corr=df.corr(method ='pearson')
corr
```

```
          A         B         C         D         E         F
A  1.000000 -0.691900  0.414978 -0.604574 -0.521985 -0.549244
B -0.691900  1.000000 -0.581516  0.487248  0.376735  0.540427
C  0.414978 -0.581516  1.000000  0.017023  0.055766 -0.175102
D -0.604574  0.487248  0.017023  1.000000  0.934396  0.681660
E -0.521985  0.376735  0.055766  0.934396  1.000000  0.647625
F -0.549244  0.540427 -0.175102  0.681660  0.647625  1.000000
```

In general, we would need to test our model. `train_test_split` is a function in `Sklearn` model selection for splitting data arrays into two subsets for train-

ing data and for testing data. With this function, you do not need to divide the dataset manually. You can use from the function `train_test_split` using the following code `sklearn.model_selection import train_test_split`. By default, Sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

Here, take note that we will need to pass in the `X` and `Y` to the function. `X` refers to the `features`, while `Y` refers to the `target` of the dataset.

```
Y_POSITION = 5
model_1_features = [i for i in range(0,Y_POSITION)]
X = df.iloc[:,model_1_features]
Y = df.iloc[:,Y_POSITION]
# create model
X_train, X_test, y_train, y_test = train_test_split(X,
↪ Y, test_size=0.20, random_state=2020)
```

## 2.1 Linear Regression

Linear regression is a basic predictive analytics technique that uses historical data to predict an output variable. It is popular for predictive modeling because it is easily understood and can be explained using plain English.

The basic idea is that if we can fit a linear regression model to observed data, we can then use the model to predict any future values. For example, let us assume that we have found from historical data that the price (P) of a house is linearly dependent upon its size (S)—in fact, we found that a house's price is exactly 90 times its size. The equation will look like this: `P = 90*S`.

With this model, we can then predict the cost of any house. If we have a house that is 1,500 square feet, we can calculate its price to be: `P = 90*1500 = $135,000`.

There are two kinds of variables in a linear regression model:

- The input or predictor variable is the variable(s) that helps predict the value of the output variable. It is commonly referred to as $X$.
- The output variable is the variable that we want to predict. It is commonly referred to as $Y$.

To estimate $Y$ using linear regression, we assume the equation:
$Y_e = \alpha + \beta X$,

where $Y_e$ is the estimated or predicted value of $Y$ based on our linear equation. Our goal is to find statistically significant values of the parameters $\alpha$ and $\beta$ that minimize the difference between $Y$ and $Y_e$. If we are able to determine the optimum values of these two parameters, then we will have the line of best fit that we can use

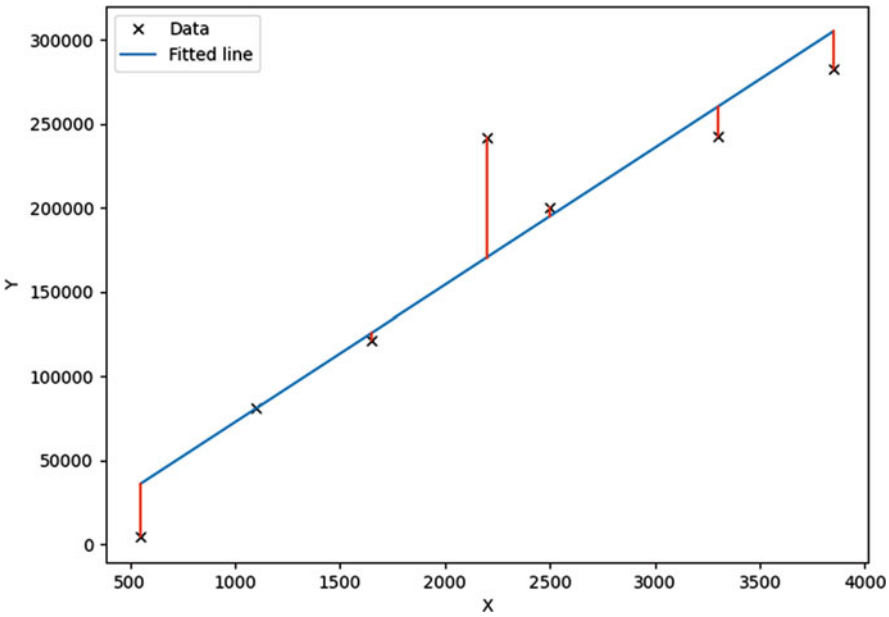



**Fig. 2.2** Regression example: house prices against house size

to predict the values of $Y$, given the value of $X$. So, how do we estimate $\alpha$ and $\beta$? We can use a method called ordinary least squares.

The objective of the least squares method is to find values of $\alpha$ and $o$ that minimize the sum of the squared difference between $Y$ and $Y_e$. Optimizing for parameters $\alpha$ and $\beta$ will provide us with fitted line shown in Fig. 2.2.

Here, we notice that when `E` increases by 1, our `Y` increases by 0.175399. Also, when `C` increases by 1, our `Y` falls by 0.044160.

```
#Model 1 : linear regression

model1 = linear_model.LinearRegression()
model1.fit(X_train, y_train)
y_pred_train1 = model1.predict(X_train)
print("Regression")
print("================================")
RMSE_train1 = mean_squared_error(y_train,y_pred_
↪train1)

print("Regression Train set: RMSE {}".format(RMSE_
↪train1))
print("================================")
```

(continues on next page)

(continued from previous page)

```
y_pred1 = model1.predict(X_test)
RMSE_test1 = mean_squared_error(y_test,y_pred1)
print("Regression Test set: RMSE {}".format(RMSE_
↪test1))
print("================================")

coef_dict = {}
for coef, feat in zip(model1.coef_,model_1_features):
    coef_dict[df.columns[feat]] = coef

print(coef_dict)
```

```
Regression
================================
Regression Train set: RMSE 0.0027616933222892287
================================
Regression Test set: RMSE 0.0042098240263563754
================================
{'A': -0.0009337757382417014, 'B': 0.
↪16012156890162915, 'C': -0.04416001542534971, 'D':␣
↪0.15217907817100398, 'E': 0.17539950794101034}
```

## 2.2 Decision Tree Regression

A decision tree is arriving at an estimate by asking a series of questions to the data, each question narrowing our possible values until the model gets confident enough to make a single prediction. The order of the question as well as their content is being determined by the model. In addition, the questions asked are all in a True/False form.

This is a little tough to grasp because it is not how humans naturally think, and perhaps the best way to show this difference is to create a real decision tree. In Fig. 2.3, we can visualize how features are utilized to make predictions for the target variable y by asking True/False questions starting at the root node and being refined at various decision nodes. The final result obtained will be the value found at the terminal node.

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees. Decision tree regression normally uses mean squared error (MSE) to decide to split a node in two or more sub-nodes. Suppose we are doing a binary tree, the algorithm first will pick a value and split the data into two subsets. For each subset, it will calculate the MSE separately. The tree chooses the value with results in smallest MSE value.

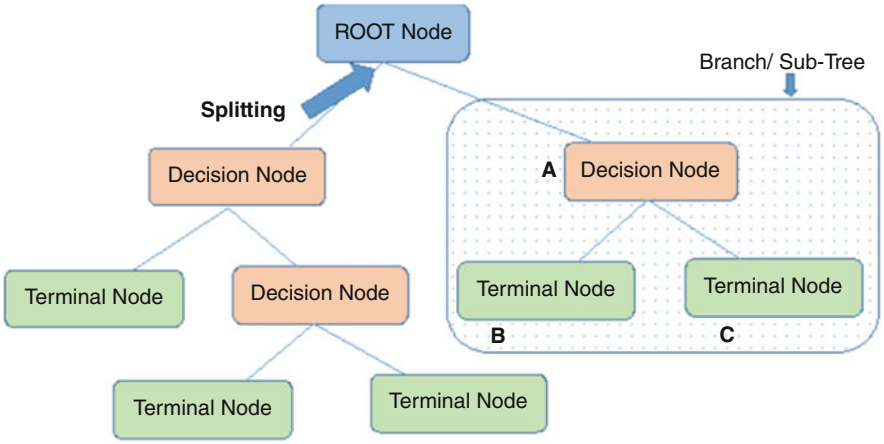



**Fig. 2.3** Splitting of a decision tree [3]

Let us examine how is splitting performed for decision tree regressors in more details. The first step to create a tree is to create the first binary decision:

1. In order to create an effective split, we need to pick a variable and the value to split on such that the two groups are as different from each other as possible.
2. For each variable, for each possible value of the possible value of that variable see whether it is better.
3. Take weighted average of two new nodes (mse*num_samples) .

To sum up, we now have:

- A single number that represents how good a split is which is the weighted average of the mean squared errors of the two groups that create.
- A way to find the best split which is to try every variable and to try every possible value of that variable and see which variable and which value gives us a split with the best score.

Training of a decision tree regressor will stop when some stopping condition is met:

1. When you hit a limit that was requested (for example: `max_depth`)
2. When your leaf nodes only have one thing in them (no further split is possible, MSE for the train will be zero but will overfit for any other set—not a useful model)

```
#Model 2 decision tree
model2 = tree.DecisionTreeRegressor()
model2.fit(X_train, y_train)
print("Decision Tree")
print("================================")
```

(continues on next page)

(continued from previous page)

```
y_pred_train2 = model2.predict(X_train)
RMSE_train2 = mean_squared_error(y_train,y_pred_
↪train2)
print("Decision Tree Train set: RMSE {}".format(RMSE_
↪train2))
print("================================")
y_pred_test2 = model2.predict(X_test)
RMSE_test2 = mean_squared_error(y_test,y_pred_test2)
print("Decision Tree Test set: RMSE {}".format(RMSE_
↪test2))
print("================================")
```

```
Decision Tree
================================
Decision Tree Train set: RMSE 1.4739259778473743e-36
================================
Decision Tree Test set: RMSE 0.008496
================================
```

## 2.3 Random Forests

The fundamental concept behind random forest is a simple but powerful one—the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (decision trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (such as stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they do not constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

So how does random forest ensure that the behavior of each individual tree is not too correlated with the behavior of any of the other trees in the model? It uses the following two methods:

1. Bagging (Bootstrap Aggregation)—Decisions trees are very sensitive to the data they are trained on—small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.
2. Feature Randomness—In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs. those in the right node. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation among the trees in the model and ultimately results in lower correlation across trees and more diversification.

As random forest is actually a collection of decision trees, this makes the algorithm slower and less effective for real-time predictions. In general, random forest can be fast to train, but quite slow to create predictions once they are trained. This is due to the fact that it has to run predictions on each individual tree and then average their predictions to create the final prediction.

Each individual tree in the random forest spits out a class prediction, and the class with the most votes becomes our model's prediction. Decision trees do suffer from overfitting, while random forest can prevent overfitting resulting in better prediction most of the time.

```
#Model 3 Random Forest
model3 = RandomForestRegressor()
model3.fit(X_train, y_train)
print("Random Forest Regressor")
print("================================")
y_pred_train3 = model3.predict(X_train)
RMSE_train3 = mean_squared_error(y_train,y_pred_
↪train3)
print("Random Forest Regressor TrainSet: RMSE {}".
↪format(RMSE_train3))
print("================================")
y_pred_test3 = model3.predict(X_test)
RMSE_test3 = mean_squared_error(y_test,y_pred_test3)
print("Random Forest Regressor TestSet: RMSE {}".
↪format(RMSE_test3))
print("================================")
```

```
Random Forest Regressor
================================
```

(continues on next page)

(continued from previous page)

```
Random Forest Regressor TrainSet: RMSE 0.
↪0004964972448979589
================================
Random Forest Regressor TestSet: RMSE 0.
↪004843255999999997
================================
```

## 2.4 Neural Network

Neural networks are the representation we make of the brain: neurons interconnected to other neurons, which forms a network. A simple information transits in a lot of them before becoming an actual thing, like "move the hand to pick up this pencil."

The operation of a neural network is straightforward: variables at each layer are taken in as inputs (for example, an image if the neural network is supposed to tell what is on an image) and produce multiple outputs (probability of whether an image is a cat).

As shown in Fig. 2.4, each blue circle represents inputs, x, and each orange circle represents a vertex where operations are performed. The vertices are connected by edges that have their own corresponding weight values, w, which is learned at training time. Each vertex will multiply the weights with the corresponding inputs

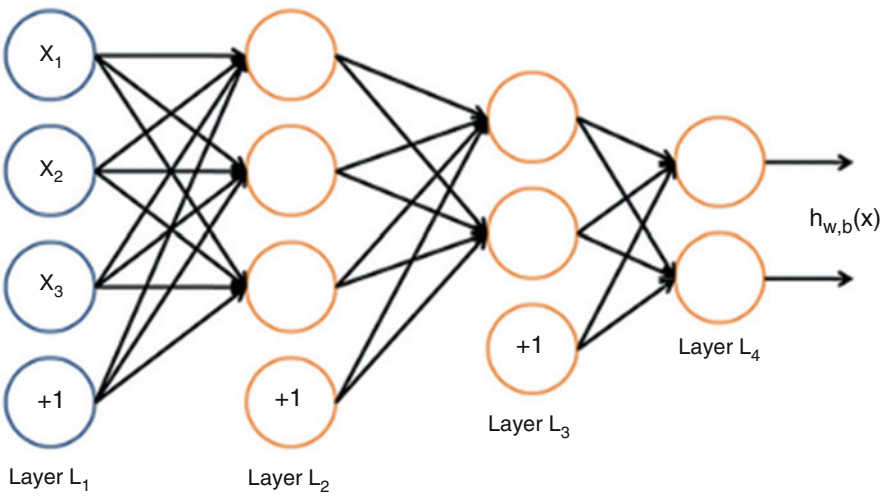



**Fig. 2.4** Example of a neural network [4]

and sum up all the values. The resulting value will be passed on to the next layer until it reaches the output.

When an input is given to the neural network, it returns an output. On the first try, it cannot get the right output by its own (except with luck) and that is why, during the learning phase, every input comes with its label, explaining what output the neural network should have guessed. If the choice is the good one, actual parameters are kept and the next input is given. However, if the obtained output does not match the label, weights are changed. Those are the only variables that can be changed during the learning phase. This process may be imagined as multiple buttons that are turned into different possibilities every time an input is not guessed correctly. To determine which weight is better to modify, a particular process, called "backpropagation," is done.

Below is the code to create a simple neural network in Python:

```
python model.add(Dense(64, input_dim=Y_position,␣
↪activation='relu'))
```

The following code is telling Python to add a layer of 64 neurons into the neural network. We can stack the models by adding more layers of neuron. Or we can simply increase the number of neurons. This can be thought of as to increase the number of "neurons" in one's brain and thereby improving one's learning ability.

```
#Model 5: neural network
print("Neural Network")
print("================================")
model = Sequential()
model.add(Dense(64, input_dim=Y_POSITION, activation=
↪'relu'))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='relu'))
# Compile mode
# https://www.tensorflow.org/guide/keras/train_and_
↪evaluate
model.compile(loss='MSE', optimizer='Adamax',␣
↪metrics=['accuracy'])
# Fit the model
model.fit(X_train, y_train, epochs=300, batch_size=5,␣
↪verbose=0)
# evaluate the model
predictions5 = model.predict(X_train)
RMSE_train5 = mean_squared_error(y_train,predictions5)
print("Neural Network TrainSet: RMSE {}".format(RMSE_
↪train5))
```

(continues on next page)

(continued from previous page)

```
print("================================")
predictions5 = model.predict(X_test)
RMSE_test5 = mean_squared_error(y_test,predictions5)
print("Neural Network TestSet: RMSE {}".format(RMSE_
↪test5))
print("================================")
```

```
Neural Network
================================
Neural Network TrainSet: RMSE 0.02496122448979592
================================
Neural Network TestSet: RMSE 0.032824
================================
```

## 2.5 Improving Regression Performance

### *2.5.1 Boxplot*

A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum," first quartile (Q1), median, third quartile (Q3), and "maximum") as shown in Fig. 2.5. It tells you about your outliers and what their values are. It can also tell you if your data are symmetrical, how tightly your data are grouped, and if and how your data are skewed.

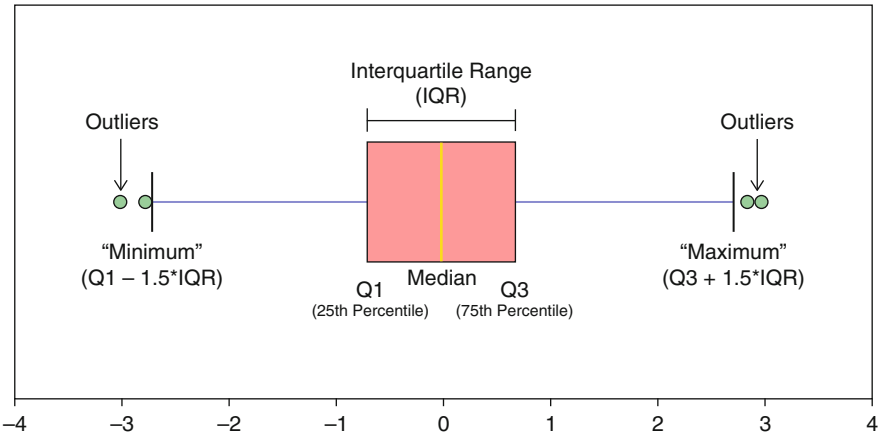



**Fig. 2.5** Example of a boxplot [2]

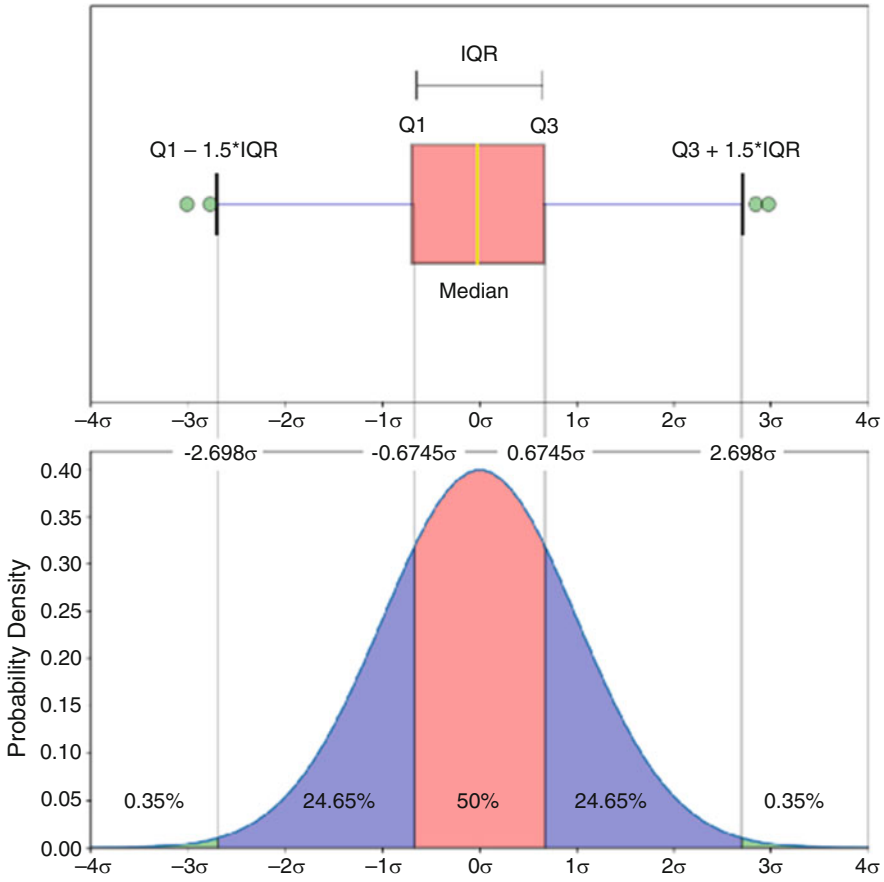



**Fig. 2.6** Example of a boxplot [2]

Figure 2.6 shows how a boxplot represents the values found on a normal distribution:

As seen, a boxplot is a great way to visualize your dataset. Now, let us try to remove the outliers using our boxplot plot. This can be easily achieved with pandas dataframe. But do note that the dataset should be numerical to do this.

Code for boxplot:

```
import seaborn as sns
import pandas as pd
boxplot = df.boxplot()
```

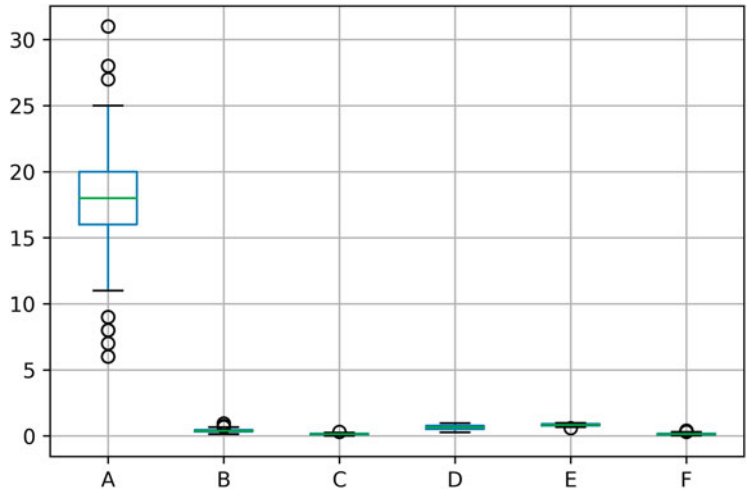



**Fig. 2.7** Outliers visualized using a boxplot

As shown in Fig. 2.7, there are values in column 0 that are outliers that are values that are extremely large or small. This can skew our dataset. A consequence of having outliers in our dataset is that our model cannot learn the right parameters. Thus, it results in a poorer prediction.

### *2.5.2 Remove Outlier*

The code below removes outlier that is more than 99th percentile. The result is shown in Fig. 2.8.

```
quantile99 = df.iloc[:,0].quantile(0.99)
df1 = df[df.iloc[:,0] < quantile99]
df1.boxplot()
```

Next, let us apply this on values lower than 1st percentile.

```
quantile1 = df.iloc[:,0].quantile(0.01)
quantile99 = df.iloc[:,0].quantile(0.99)
df2 = df[(df.iloc[:,0] > quantile1) & (df.iloc[:,0] <␣
↪quantile99)]
df2.boxplot()
```

As shown in Fig. 2.9, we have removed the 99th percentile outliers from the data successfully.

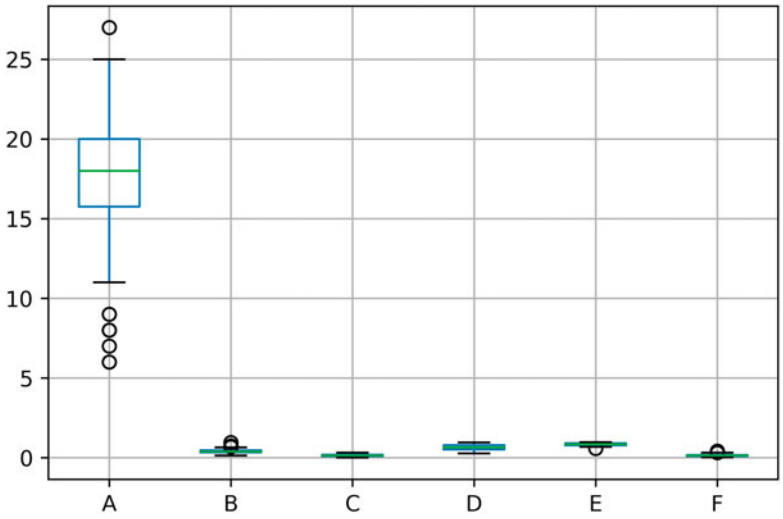



**Fig. 2.8** Boxplot after 99th percentile outliers removed

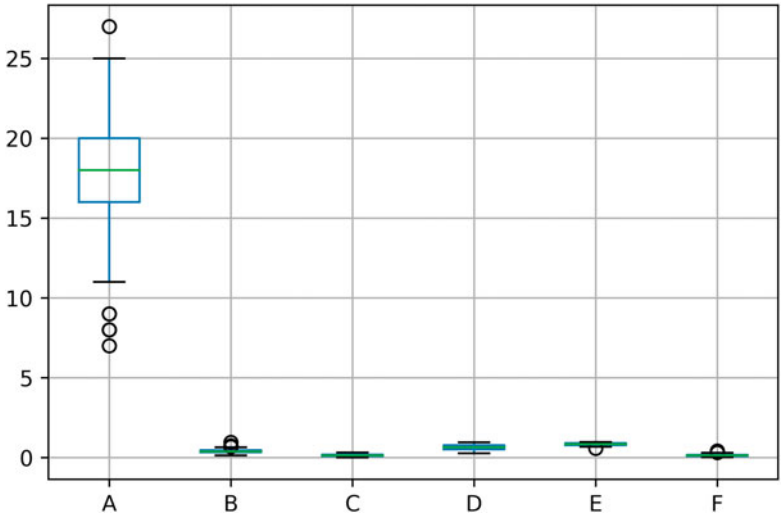



**Fig. 2.9** Boxplot after 1st and 99th percentile outliers removed

```
df2.shape
```

```
(118, 6)
```

### 2.5.3 Remove NA

To drop all the rows with the NaN values, you may use:

```
df.dropna()
```

```
df1 = df1.dropna()
```

### 2.5.4 Feature Importance

Apart from cleaning, we can apply use variables that we deem to be important to us. One way of doing so is via feature importance of random forest trees. In many use cases, it is equally important to not only have an accurate, but also an interpretable model. Oftentimes, apart from wanting to know what our model's house price prediction is, we also wonder why it is this high/low and which features are most important in determining the forecast. Another example might be predicting customer churn—it is very nice to have a model that is successfully predicting which customers are prone to churn, but identifying which variables are important can help us in early detection and maybe even improving the product/service.

Knowing feature importance indicated by machine learning models can benefit you in multiple ways, for example:

1. By getting a better understanding of the model's logic, you cannot only verify it being correct but also work on improving the model by focusing only on the important variables.
2. The above can be used for variable selection—you can remove x variables that are not that significant and have similar or better performance in much shorter training time.
3. In some business cases, it makes sense to sacrifice some accuracy for the sake of interpretability. For example, when a bank rejects a loan application, it must also have a reasoning behind the decision, which can also be presented to the customer.

We can obtain the feature importance using this code:

```
importances = RF.feature_importances_
```

Then, we can sort the feature importance for ranking and indexing.

```
indices = numpy.argsort(importances)[::-1]
```

```
import numpy
RF = model3
importances = RF.feature_importances_
std = numpy.std([tree.feature_importances_ for tree␣
↪in RF.estimators_],axis=0)
indices = numpy.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature (Column index) %s (%f)" % (f +␣
↪1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature (Column index) 3 (0.346682)
2. feature (Column index) 1 (0.217437)
3. feature (Column index) 0 (0.174081)
4. feature (Column index) 4 (0.172636)
5. feature (Column index) 2 (0.089163)
```

Let us use the top 3 features and retrain another model. Here, we took a shorter time to train the model, yet the RMSE does not suffer due to fewer features.

```
indices_top3 = indices[:3]
print(indices_top3)
dataset=df
df = pd.DataFrame(df)

Y_position = 5
TOP_N_FEATURE = 3

X = dataset.iloc[:,indices_top3]
Y = dataset.iloc[:,Y_position]
# create model
X_train, X_test, y_train, y_test = train_test_split(X,
↪ Y, test_size=0.20, random_state=2020)
```

(continues on next page)

(continued from previous page)

```
#Model 1 : linear regression


model1 = linear_model.LinearRegression()
model1.fit(X_train, y_train)
y_pred_train1 = model1.predict(X_train)
print("Regression")
print("================================")
RMSE_train1 = mean_squared_error(y_train,y_pred_
↪train1)

print("Regression TrainSet: RMSE {}".format(RMSE_
↪train1))
print("================================")
y_pred1 = model1.predict(X_test)
RMSE_test1 = mean_squared_error(y_test,y_pred1)
print("Regression Testset: RMSE {}".format(RMSE_
↪test1))
print("================================")
```

```
[3 1 0]
Regression
================================
Regression TrainSet: RMSE 0.0027952079052752685
================================
Regression Testset: RMSE 0.004341758028139643
================================
```

## Exercises

We will utilize this dataset [1]:
https://www.kaggle.com/datasets/ashydv/advertising-dataset

The advertising dataset contains the amount of sales of a product, in thousands of units together with the advertising expenditure in thousands of dollars, across TV, radio, and newspapers.

In this exercise, try to predict the amount of sales generated based on the advertising budget allocated to the various advertising mediums.

Here you can put what you have learned into practice by creating, testing, and tuning the performance of various regression models.

```
import pandas as pd
df = pd.read_csv('advertising.csv')
df.head()
```

```
TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

## References


1. Ashish (2019) Advertising dataset. https://www.kaggle.com/datasets/ashydv/advertising-dataset
2. Galarnyk M (2019) Understanding boxplots - KDnuggets. In: KDnuggets. https://www.kdnuggets.com/understanding-boxplots.html. Accessed 27 Apr 2023
3. Nicholson C (2023) Decision tree. In: Pathmind. http://wiki.pathmind.com/decision-tree. Accessed 27 Apr 2023
4. Stanford University (2013) Unsupervised feature learning and deep learning tutorial. In: Unsupervised Feature Learning and Deep Learning Tutorial. http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/. Accessed 27 Apr 2023
5. Vigen T (2013) Spurious correlations. In: Spurious Correlations. http://tylervigen.com/spurious-correlations. Accessed 27 Apr 2023

# Chapter 3
# Classification


**Learning Outcomes**

- Learn the difference between classification and regression. Be able to differentiate between classification and regression problems.
- Learn and apply basic models for classification tasks using sklearn and keras.
- Learn data processing techniques to achieve better classification results.

We learned about regression previously. Now, let us take a look at classification. Fundamentally, classification is about predicting a label and regression is about predicting a quantity.

Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). The output variables are often called labels or categories. The mapping function predicts the class or category for a given observation.

For example, an email of text can be classified as belonging to one of two classes: "spam" and "not spam." A classification can have real-valued or discrete input variables.

Here are different types of classification problem:

- A problem with two classes is often called a two-class or binary classification problem.
- A problem with more than two classes is often called a multi-class classification problem.
- A problem where an example is assigned multiple classes is called a multi-label classification problem.

It is common for classification models to predict a continuous value as the probability of a given example belonging to each output class. The probabilities can be interpreted as the likelihood or confidence of a given example belonging to



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_3

each class. A predicted probability can be converted into a class value by selecting the class label that has the highest probability.

For example, a specific email of text may be assigned the probabilities of 0.1 as being "spam" and 0.9 as being "not spam." We can convert these probabilities to a class label by selecting the "not spam" label as it has the highest predicted likelihood.

There are many ways to estimate the skill of a classification predictive model, but perhaps the most common is to calculate the classification accuracy.

The classification accuracy is the percentage of correctly classified examples out of all predictions made.

For example, if a classification predictive model made 5 predictions and 3 of them were correct and 2 of them were incorrect, then the classification accuracy of the model based on just these predictions would be:

```
accuracy = correct predictions / total predictions *␣
↪100
accuracy = 3 / 5 * 100
accuracy = 60%
```

An algorithm that is capable of learning a classification predictive model is called a classification algorithm.

Dataset: "Diabetes (Edited).csv"

You can obtain the dataset from this link https://www.dropbox.com/s/ggxo241uog06yhj/Diabetes (Edited).csv?dl=0

Also, you may run the following code in order to download the dataset in `google colab`:

```
!wget https://www.dropbox.com/s/ggxo241uog06yhj/
↪Diabetes%20%28Edited%29.csv?dl=0 -O --quiet
↪"Diabetes (Edited).csv"
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report,␣
↪confusion_matrix
from sklearn.model_selection import train_test_split
import numpy
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier,
↪GradientBoostingClassifier
import pandas as pd
import csv
```

```
Using TensorFlow backend.
```

First, we will work on preprocessing the data. For numerical data, often we would preprocess the data by scaling it. In our example, we apply standard scalar, a popular preprocessing technique.

Standardization is a transformation that centers the data by removing the mean value of each feature and then scaling it by dividing (non-constant) features by their standard deviation. After standardizing data, the mean will be zero and the standard deviation one.

Standardization can drastically improve the performance of models. For instance, many elements used in the objective function of a learning algorithm assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Here the code that does the scaling is as follows:

```
scaler = preprocessing.StandardScaler().fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

Notice that we are using the scalar fitted on our `X_train` to transform values in X_test. This is to ensure that our model does not learn from the testing data. Usually, we would split our data before applying scaling. It is a bad practice to do scaling on the full dataset.

Apart from standard scaling, we can use other scalar such as `MinMaxScalar`. `feature_range` refers to the highest and lowest values after scaling. By default, "feature_range" is −1 to 1. However, this range may prove to be too small as changes in our variable would be compressed to maximum of −1 to 1.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(-3,3))
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

```
Y_position = 8

# fix random seed for reproducibility
numpy.random.seed(7)

df = pd.read_csv('Diabetes (Edited).csv')
print(df)
# summary statistics
```

(continues on next page)

(continued from previous page)

```
print(df.describe())

X = df.iloc[:,0:Y_position]
Y = df.iloc[:,Y_position]

# create model
X_train, X_test, y_train, y_test = train_test_split(X,
↪ Y, test_size=0.40, random_state=2020)

#scaling to around -2 to 2 (Z)
scaler = preprocessing.StandardScaler().fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

```
      A    B   C   D    E     F      G   H  I
0     6  148  72  35    0  33.6  0.627  50  1
1     1   85  66  29    0  26.6  0.351  31  0
2     8  183  64   0    0  23.3  0.672  32  1
3     1   89  66  23   94  28.1  0.167  21  0
4     0  137  40  35  168  43.1  2.288  33  1
..   ..  ...  ..  ..  ...   ...    ...  .. ..
763  10  101  76  48  180  32.9  0.171  63  0
764   2  122  70  27    0  36.8  0.340  27  0
765   5  121  72  23  112  26.2  0.245  30  0
766   1  126  60   0    0  30.1  0.349  47  1
767   1   93  70  31    0  30.4  0.315  23  0

[768 rows x 9 columns]
                A           B           C           D           E           F  \
count  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000
mean     3.845052  120.894531   69.105469   20.536458   79.799479   31.992578
std      3.369578   31.972618   19.355807   15.952218  115.244002    7.884160
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      1.000000   99.000000   62.000000    0.000000    0.000000   27.300000
50%      3.000000  117.000000   72.000000   23.000000   30.500000   32.000000
75%      6.000000  140.250000   80.000000   32.000000  127.250000   36.600000
max     17.000000  199.000000  122.000000   99.000000  846.000000   67.100000

                G           H           I
count  768.000000  768.000000  768.000000
mean     0.471876   33.240885    0.348958
std      0.331329   11.760232    0.476951
min      0.078000   21.000000    0.000000
25%      0.243750   24.000000    0.000000
50%      0.372500   29.000000    0.000000
75%      0.626250   41.000000    1.000000
max      2.420000   81.000000    1.000000
```

In order to reduce code duplication as seen in the chapter on `Regression`, we can abstract the model and create a function to help us train and predict. Here is the explanation for the code:

```
model.fit(scaled_X_train, y_train)
```

We train the model using `scaled_X_train` and provide its label `y_train`

```
y_predicted = model3.predict(scaled_X_test)
```

We predict the model on our testing data and store its result in the variable `y_predicted`

```
cm_test = confusion_matrix(y_test,y_pred)
```

We create a confusion matrix given our y_test and y_pred. And what is a confusion matrix?

A confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making:

- Expected down the side: Each row of the matrix corresponds to a predicted class.
- Predicted across the top: Each column of the matrix corresponds to an actual class.

```
acc_test = (cm_test[0,0] + cm_test[1,1]) /␣
↪sum(sum(cm_test))
```

Lastly, this code calculates the accuracy for us. Accuracy is the number of correctly predicted data points out of all the data points. More formally, it is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives. These values are the outputs of a confusion matrix.

Here, we are assuming a binary classification problem. For multi-class classification problem, I would highly recommend using sklearn's `accuracy` function for its calculation.

```
def train_and_predict_using_model(model_name= "",
↪model=None):
    model.fit(scaled_X_train, y_train)
    y_pred_train = model.predict(scaled_X_train)
    cm_train = confusion_matrix(y_train,y_pred_train)
    print(model_name)
```

(continues on next page)

(continued from previous page)

```
    print("================================")
    print("Training confusion matrix: ")
    print(cm_train)
    acc_train = (cm_train[0,0] + cm_train[1,1]) /␣
↪sum(sum(cm_train))
    print("TrainSet: Accuracy %.2f%%" % (acc_
↪train*100))
    print("================================")
    y_pred = model.predict(scaled_X_test)
    cm_test = confusion_matrix(y_test,y_pred)
    print(cm_test)
    acc_test = (cm_test[0,0] + cm_test[1,1]) /␣
↪sum(sum(cm_test))
    print("Testset: Accuracy %.2f%%" % (acc_test*100))
    print("================================")
```

## 3.1 Logistic Regression

Why not use linear regression?

Let us say we have some tumor size malignancy with values shown by the plotted X's. As a classification issue, plotting shows that all values fall between 0 and 1. From the plot, we can see that fitting the best regression line with the threshold set at 0.5 works well to classify our data [2]. As shown in Fig. 3.1, we are able to select a point on the x-axis where all numbers to its left are negative and those to its right are positive.

However, things would start to become messy if there are outliers in the data. In Fig. 3.2, we illustrate what would happen if there were multiple outliers with a tumor size of 14 cm.

In the above example, if we keep the original threshold of 0.5, the best found regression line will not be good enough to help differentiate between the two classes. It will insert some samples from the positive class into the negative class. The initial decision boundary that distinguished between malignant and benign tumors would have resulted in misclassifications. When outliers are present, using linear regression in this way might lead to skewed results. As a result, for classification jobs, we often employ logistic regression instead.

As pointed out previously, logistic regression employs the sigmoid function, which is more resistant to outliers in classification tasks. The logistic function is a sigmoid function that accepts any real value between zero and one and is used to

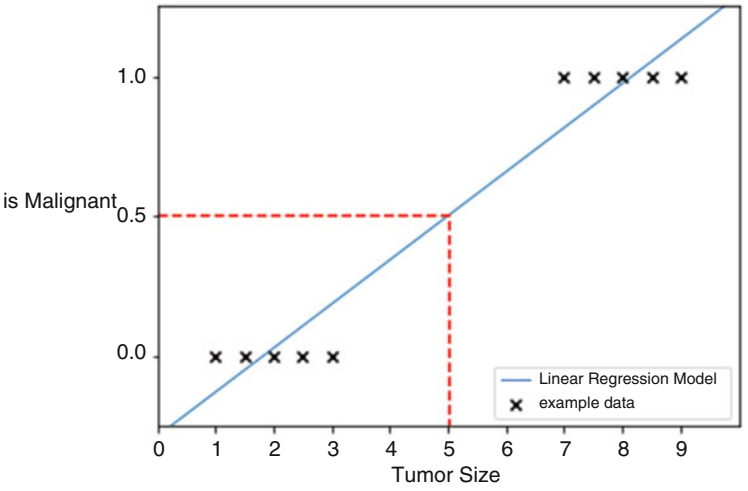



**Fig. 3.1** Regression for classification

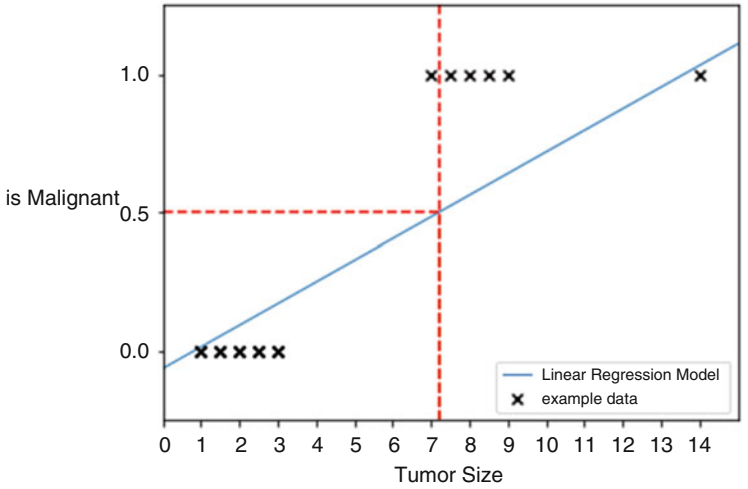



**Fig. 3.2** Problem using regression to classify when outliers are present

regress between binary classes in logistic regression. It is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

And if we plot it, the graph drawn will be in the shape of an S curve as shown in Fig. 3.3.

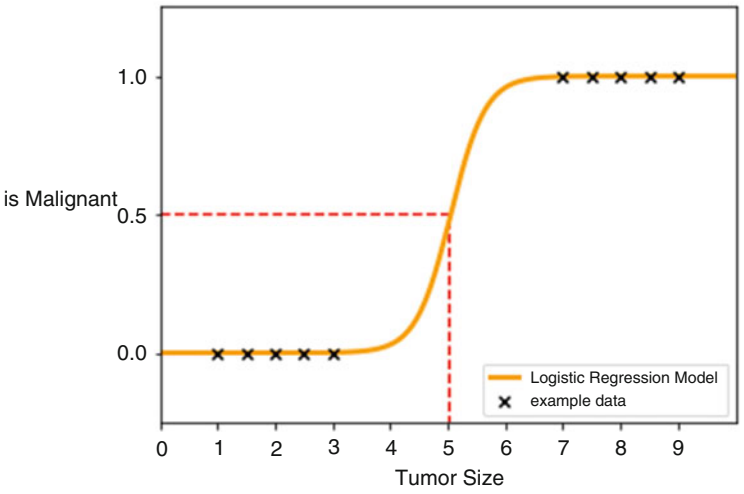



**Fig. 3.3** Logistic regression for classification

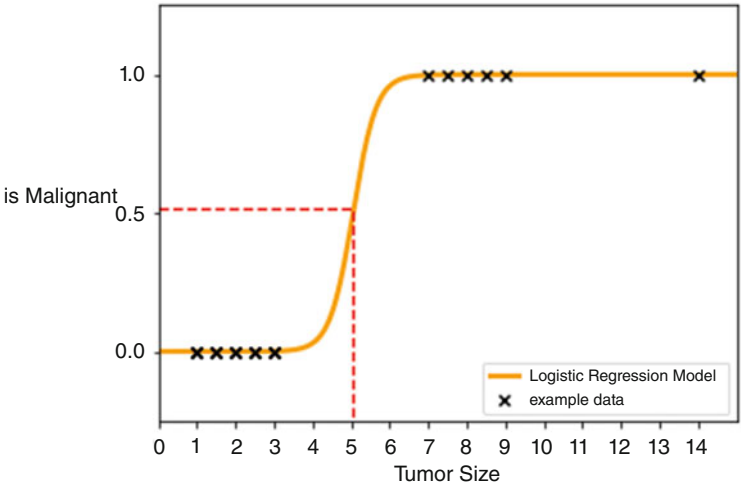



**Fig. 3.4** Logistic regression for classification with outliers

As you can see in Fig. 3.4, the logistic regression model is more robust to outliers in a binary classification task compared to linear regression.

Take the scenario in which we are attempting to solve a classification issue and the ideal result would be a probability. Our probability output would range between the values of 0 to 1, representing the probability that the event occurred. However, applying linear regression would provide results ranging from 1 to infinity. On the other hand, when data points are mapped to a `sigmoid` function, they are mapped between 0 and 1, based on the linear regression outcome.

```
#https://scikit-learn.org/stable/modules/generated/
↪sklearn.linear_model.LogisticRegression.html
linear_classifier = linear_model.
↪LogisticRegression(random_state=123)
linear_classifier.fit(scaled_X_train, y_train)
y_pred_train1 = linear_classifier.predict(scaled_X_
↪train)
cm1_train = confusion_matrix(y_train,y_pred_train1)
print("Regression")
print("================================")
print(cm1_train)
acc_train1 = (cm1_train[0,0] + cm1_train[1,1]) /␣
↪sum(sum(cm1_train))
print("Regression TrainSet: Accuracy %.2f%%" % (acc_
↪train1*100))
print("================================")
y_pred1 = linear_classifier.predict(scaled_X_test)
cm1 = confusion_matrix(y_test,y_pred1)
print(cm1)
acc1 = (cm1[0,0] + cm1[1,1]) / sum(sum(cm1))
print("Regression Testset: Accuracy %.2f%%" %␣
↪(acc1*100))
print("================================")
```

```
Regression
================================
[[274  31]
 [ 62  93]]
Regression TrainSet: Accuracy 79.78%
================================
[[172  23]
 [ 53  60]]
Regression Testset: Accuracy 75.32%
================================
```

Sample result:

```
================================
[[274  31]
 [ 62  93]]
Regression TrainSet: Accuracy 79.78%
================================
```

```
[[274  31]
 [ 62  93]]
```

Here is an example of a confusion matrix in Python:

- 274 is when the actual class is true and predicted class is true.
- 31 is when the actual class is true and predicted class is false.
- 62 is when the actual class is false and predicted class is true.
- 93 is when the actual class is false and predicted class is false.

**Improvement to Our Code**

Recall we have written a helper function to help us to capture the logic of training the model, predicting the output and printing the train and test accuracy as well as confusion matrix? Let us put it to use here!

```
train_and_predict_using_model('Logistic Regression',␣
↪linear_classifier)
```

```
Logistic Regression
================================
Training confusion matrix:
[[274  31]
 [ 62  93]]
TrainSet: Accuracy 79.78%
================================
[[172  23]
 [ 53  60]]
Testset: Accuracy 75.32%
================================
```

We have managed to reduce multiple lines of code to a succinct function call. This is a huge improvement in terms of code maintenance and code changes. If we need to change any of our code, we only have to apply it on our `train_and_predict_using_model` function.

## 3.2 Decision Tree and Random Forest

The code and intuition behind decision tree and random forest is similar to that in regression. Thus, we will not be delving deeper into both models.

The code is as follows:

```
decision_tree_clf = tree.DecisionTreeClassifier()
train_and_predict_using_model('Decision Tree␣
↪Classifier', linear_classifier)

print(
    '\n\n'
)

rf_clf = RandomForestClassifier(n_estimators=100, max_
↪depth=2,random_state=0)
train_and_predict_using_model('Random Forest␣
↪Classifier', rf_clf)
```

```
Decision Tree Classifier
================================
Training confusion matrix:
[[274  31]
 [ 62  93]]
TrainSet: Accuracy 79.78%
================================
[[172  23]
 [ 53  60]]
Testset: Accuracy 75.32%
================================



Random Forest Classifier
================================
Training confusion matrix:
[[290  15]
 [ 96  59]]
TrainSet: Accuracy 75.87%
================================
[[184  11]
 [ 80  33]]
Testset: Accuracy 70.45%
================================
```

## 3.3 Neural Network

In this section, we will introduce the neural network. Similar to logistic regression, we have to map our output from `-inf` to `inf` into `0` to `1`. Here, we will have to add a `Dense` layer with a `sigmoid` activation function. For multi-class, we should use a `softmax` activation function.

```
model.add(Dense(1, activation='sigmoid'))
```

Here, we added a last layer mapping to a sigmoid function. Notice that we have 1 neuron in this layer as we would like to have 1 prediction. This might be different for multi-class, and we should always check out the documentation.

```
model.compile(loss='binary_crossentropy', optimizer=
↪'Adamax', metrics=['accuracy'])
```

Also, we would need to tell the model that we need to use a different loss function. Here, for binary problem (Yes/No). `binary_crossentropy` is the way to go. For multi-class problem, we might need to use `categorical_crossentropy` as the loss function.

```
#Neural network
#https://www.tensorflow.org/guide/keras/train_and_
↪evaluate
model = Sequential()
model.add(Dense(5, input_dim=Y_position, activation=
↪'relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile model
# https://www.tensorflow.org/guide/keras/train_and_
↪evaluate
model.compile(loss='binary_crossentropy', optimizer=
↪'Adamax', metrics=['accuracy'])

# Fit the model
model.fit(scaled_X_train, y_train, epochs=1, batch_
↪size=20, verbose=0)

# evaluate the model
scores = model.evaluate(scaled_X_train, y_train)

print("Neural Network Trainset: \n%s: %.2f%%" %␣
↪(model.metrics_names[1], scores[1]*100))
```

(continues on next page)

(continued from previous page)

```
predictions = model.predict(scaled_X_test)

y_pred = (predictions > 0.5)
y_pred = y_pred*1 #convert to 0,1 instead of True␣
↪False
cm = confusion_matrix(y_test, y_pred)
print("====================================")
print("====================================")
print("Neural Network on testset confusion matrix")
print(cm)

## Get accurary from Confusion matrix
## Position 0,0 and 1,1 are the correct predictions
acc = (cm[0,0] + cm[1,1]) / sum(sum(cm))
print("Neural Network on TestSet: Accuracy %.2f%%" %␣
↪(acc*100))
```

```
460/460 [==============================] - 0s 63us/
↪step
Neural Network Trainset:
accuracy: 71.09%
====================================
====================================
Neural Network on testset confusion matrix
[[177  18]
 [ 78  35]]
Neural Network on TestSet: Accuracy 68.83%
```

From above, notice that the training accuracy is at 71%, which might be a case of underfitting. Underfitting happens when our trained model is too simple to capture the complexity of the underlying data distribution. To improve our model, we can always increase the number of neurons / layer or increase the epoch for training.

```
#Neural network
#https://www.tensorflow.org/guide/keras/train_and_
↪evaluate
model = Sequential()
model.add(Dense(10, input_dim=Y_position, activation=
↪'relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(256, activation='tanh'))
```

(continues on next page)

(continued from previous page)

```
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))

# Compile model
# https://www.tensorflow.org/guide/keras/train_and_
↪evaluate
model.compile(loss='binary_crossentropy', optimizer=
↪'RMSprop', metrics=['accuracy'])

# Fit the model
model.fit(scaled_X_train, y_train, epochs=200, batch_
↪size=20, verbose=0)

# evaluate the model
scores = model.evaluate(scaled_X_train, y_train)

print("Neural Network Trainset: \n%s: %.2f%%" %␣
↪(model.metrics_names[1], scores[1]*100))

predictions = model.predict(scaled_X_test)

y_pred = (predictions > 0.5)
y_pred = y_pred*1 #convert to 0,1 instead of True␣
↪False
cm = confusion_matrix(y_test, y_pred)
print("==================================")
print("==================================")
print("Neural Network on testset confusion matrix")
print(cm)

## Get accuracy from Confusion matrix
## Position 0,0 and 1,1 are the correct predictions
acc = (cm[0,0] + cm[1,1]) / sum(sum(cm))
print("Neural Network on TestSet: Accuracy %.2f%%" %␣
↪(acc*100))
```

```
460/460 [==============================] - 0s 126us/
↪step
Neural Network Trainset:
accuracy: 99.57%
==================================
==================================
```

(continues on next page)

(continued from previous page)

```
Neural Network on testset confusion matrix
[[152  43]
 [ 43  70]]
Neural Network on TestSet: Accuracy 72.08%
```

Now, our accuracy on training has reached 99%. However, accuracy of test is still lower. This might be because of testing dataset differing from training dataset or overfitting. For overfitting, we will look at some regularization techniques. For now, adding `Dropout` layer and reducing training `epoch` would work just fine.

## 3.4 Support Vector Machines

The support vector machine is a widely used machine learning algorithm, often applied on classification, but sometimes also for regression and outlier detection tasks. The support vector machine algorithm aims to find a suitable hyperplane in a multidimensional space that can separate the data into two distinct classes.

However, there are many possible hyperplanes that can fit the above-mentioned criteria. Hence, instead of simply selecting a suitable hyperplane, the support vector machine algorithm is made to select the best plane. This is achieved by finding a plane that maximizes the distance between both classes. The data points that determine the position of this hyperplane are called support vectors, and they lie closest to the resulting hyperplane.

```
from sklearn import svm

clf = svm.SVC()
train_and_predict_using_model("SVM (Classifier)", clf)
```

```
SVM (Classifier)
================================
Training confusion matrix:
[[361  46]
 [101 106]]
TrainSet: Accuracy 76.06%
================================
[[84  9]
 [31 30]]
Testset: Accuracy 74.03%
================================
```

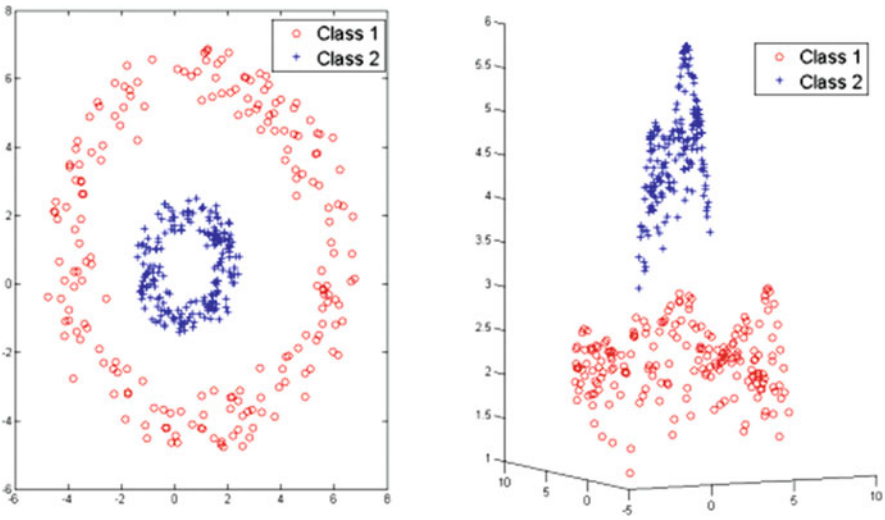



**Fig. 3.5** Support vector machine using a radial basis function kernel

### *3.4.1 Important Hyperparameters*

For SVM here are some important parameters to take note of:

**Kernel**
Kernel functions generally transform the data so that a non-linear decision surface is transformed into a linear one in a higher number of dimension spaces. Some of the possible kernel functions are as follows:

- Radial basis function (rbf)
- Polynomial
- Sigmoid

Here is an illustrated use of a rbf kernel in Fig. 3.5. The rbf kernel maps the data into a higher dimensional space by calculating the dot products and squares against data in the training set. After mapping into this higher dimensional space, we can then apply SVM for the final linear classification.

```
rbf_svc = svm.SVC(kernel='rbf')
train_and_predict_using_model("SVM (RBF kernel)", rbf_
↪svc)
```

```
SVM (RBF kernel)
================================
Training confusion matrix:
```

(continues on next page)

(continued from previous page)

```
[[361  46]
 [101 106]]
TrainSet: Accuracy 76.06%
================================
[[84  9]
 [31 30]]
Testset: Accuracy 74.03%
================================
```

```
rbf_svc = svm.SVC(kernel='poly')
train_and_predict_using_model("SVM (polynomial kernel)
↪", rbf_svc)
```

```
SVM (polynomial kernel)
================================
Training confusion matrix:
[[393  14]
 [148  59]]
TrainSet: Accuracy 73.62%
================================
[[89  4]
 [47 14]]
Testset: Accuracy 66.88%
================================
```

```
rbf_svc = svm.SVC(kernel='sigmoid')
train_and_predict_using_model("SVM (sigmoid kernel)",␣
↪rbf_svc)
```

```
SVM (sigmoid kernel)
================================
Training confusion matrix:
[[320  87]
 [112  95]]
TrainSet: Accuracy 67.59%
================================
[[68 25]
 [35 26]]
Testset: Accuracy 61.04%
================================
```

Another important parameter would be class_weight. Here, it is mainly used for imbalanced datasets.

```
# fit the model and get the separating hyperplane␣
↪using weighted classes
wclf = svm.SVC(kernel='linear', class_weight={1:2})
train_and_predict_using_model('SVM uneven class weight
↪', wclf)
```

```
SVM uneven class weight
================================
Training confusion matrix:
[[316  91]
 [ 75 132]]
TrainSet: Accuracy 72.96%
================================
[[71 22]
 [18 43]]
Testset: Accuracy 74.03%
================================
```

## 3.5 Naive Bayes

Naive Bayes is a classification technique based on Bayes' theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as "Naive."

Naive Bayes model is easy to build and particularly useful for very large datasets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability `P(c|x)` from `P(c)`, `P(x)`, and `P(x|c)` as shown in the equation below:

$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$

```
from sklearn.naive_bayes import GaussianNB

# maximum likelihood
```

(continues on next page)

(continued from previous page)

```
gnb = GaussianNB()
train_and_predict_using_model("Naive Bayes", gnb)
```

```
Naive Bayes
================================
Training confusion matrix:
[[337  70]
 [ 93 114]]
TrainSet: Accuracy 73.45%
================================
[[78 15]
 [28 33]]
Testset: Accuracy 72.08%
================================
```

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.naive_bayes import GaussianNB

X, y = make_classification(n_samples=1000, weights=[0.
↪1, 0.9])
# your GNB estimator
gnb = GaussianNB()
gnb.fit(X, y)

print("model prior {} close to your defined prior of
↪{}".format(gnb.class_prior_, [0.1,0.9]))
```

```
model prior [0.105 0.895] close to your defined prior␣
↪of [0.1, 0.9]
```

## 3.6 Improving Classification Performance

Similar to how we utilized feature importance for regression tasks, we can apply feature importance to classification tasks as well.

```
RF = model3
importances = RF.feature_importances_
```

(continues on next page)

(continued from previous page)

```
std = numpy.std([tree.feature_importances_ for tree␣
↪in RF.estimators_],
axis=0)
indices = numpy.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
print("%d. feature (Column index) %s (%f)" % (f + 1,␣
↪indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature (Column index) 1 (0.307004)
2. feature (Column index) 7 (0.237150)
3. feature (Column index) 0 (0.129340)
4. feature (Column index) 5 (0.129255)
5. feature (Column index) 6 (0.069927)
6. feature (Column index) 4 (0.055137)
7. feature (Column index) 2 (0.044458)
8. feature (Column index) 3 (0.027729)
```

```
indices_top3 = indices[:3]
print(indices_top3)

# fix random seed for reproducibility
numpy.random.seed(7)
# load pima indians dataset
dataset = numpy.loadtxt("Diabetes (Edited).csv",␣
↪delimiter=",")

df = pd.DataFrame(dataset)

Y_position = 8
TOP_N_FEATURE = 3

X = dataset[:,indices_top3]
Y = dataset[:,Y_position]
# create model
X_train, X_test, y_train, y_test = train_test_split(X,
↪ Y, test_size=0.20, random_state=2020)
```

(continues on next page)

(continued from previous page)

```
#scaling to around -2 to 2 (Z)
scaler = preprocessing.StandardScaler().fit(X_train)
scaled_X_train = scaler.transform(X_train)
scaled_X_test = scaler.transform(X_test)


#Model 1 : linear regression
linear_classifier = linear_model.
↪LogisticRegression(random_state=123)
linear_classifier.fit(scaled_X_train, y_train)
y_pred_train1 = linear_classifier.predict(scaled_X_
↪train)
cm1_train = confusion_matrix(y_train,y_pred_train1)
print("Regression")
print("================================")
print(cm1_train)
acc_train1 = (cm1_train[0,0] + cm1_train[1,1]) /␣
↪sum(sum(cm1_train))
print("Regression TrainSet: Accuracy %.2f%%" % (acc_
↪train1*100))
print("================================")
y_pred1 = linear_classifier.predict(scaled_X_test)
cm1 = confusion_matrix(y_test,y_pred1)
print(cm1)
acc1 = (cm1[0,0] + cm1[1,1]) / sum(sum(cm1))
print("Regression Testset: Accuracy %.2f%%" %␣
↪(acc1*100))
print("================================")
print("================================")
print("================================")


#Model 2: decision tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(scaled_X_train, y_train)
y_pred_train2 = clf.predict(scaled_X_train)
cm2_train = confusion_matrix(y_train,y_pred_train2)
print("Decision Tree")
print("================================")
print(cm2_train)
acc_train2 = (cm2_train[0,0] + cm2_train[1,1]) /␣
↪sum(sum(cm2_train))
```

(continues on next page)

(continued from previous page)

```
print("Decsion Tree TrainSet: Accuracy %.2f%%" % (acc_
↪train2*100))
print("================================")
y_pred2 = clf.predict(scaled_X_test)
cm2 = confusion_matrix(y_test,y_pred2)
acc2 = (cm2[0,0] + cm2[1,1]) / sum(sum(cm2))
print(cm2)
print("Decision Tree Testset: Accuracy %.2f%%" %␣
↪(acc2*100))
print("================================")
print("================================")
print("================================")




#Model 3 random forest
model3 = RandomForestClassifier(n_estimators=100, max_
↪depth=2,random_state=0)
model3.fit(scaled_X_train, y_train)
y_predicted3 = model3.predict(scaled_X_test)

y_pred_train3 = model3.predict(scaled_X_train)
cm3_train = confusion_matrix(y_train,y_pred_train3)
print("Random Forest")
print("================================")
print(cm3_train)
acc_train3 = (cm3_train[0,0] + cm3_train[1,1]) /␣
↪sum(sum(cm3_train))
print("Random Forest TrainSet: Accuracy %.2f%%" %␣
↪(acc_train3*100))
print("================================")
y_pred3 = model3.predict(scaled_X_test)
cm_test3 = confusion_matrix(y_test,y_pred3)
print(cm_test3)
acc_test3 = (cm_test3[0,0] + cm_test3[1,1]) /␣
↪sum(sum(cm_test3))
print("Random Forest Testset: Accuracy %.2f%%" % (acc_
↪test3*100))
print("================================")
print("================================")
print("================================")
```

(continues on next page)

(continued from previous page)

```
#Model 4: XGBoost


print("Xgboost")
print("================================")


model4 = GradientBoostingClassifier(random_state=0)
model4.fit(scaled_X_train, y_train)
y_pred_train4 = model4.predict(scaled_X_train)
cm4_train = confusion_matrix(y_train,y_pred_train4)
print(cm4_train)
acc_train4 = (cm4_train[0,0] + cm4_train[1,1]) /␣
↪sum(sum(cm4_train))
print("Xgboost TrainSet: Accuracy %.2f%%" % (acc_
↪train4*100))
predictions = model4.predict(scaled_X_test)
y_pred4 = (predictions > 0.5)
y_pred4 =y_pred4*1 #convert to 0,1 instead of True␣
↪False
cm4 = confusion_matrix(y_test, y_pred4)
print("==================================")
print("Xgboost on testset confusion matrix")
print(cm4)
acc4 = (cm4[0,0] + cm4[1,1]) / sum(sum(cm4))
print("Xgboost on TestSet: Accuracy %.2f%%" %␣
↪(acc4*100))
print("==================================")

#Model 5: neural network
model = Sequential()
model.add(Dense(10, input_dim=TOP_N_FEATURE,␣
↪activation='relu'))
#model.add(Dense(10, activation='relu'))
#model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
# Compile mode
# https://www.tensorflow.org/guide/keras/train_and_
↪evaluate

model.compile(loss='binary_crossentropy', optimizer=
↪'Adamax', metrics=['accuracy'])
```

(continues on next page)

(continued from previous page)

```
# Fit the model
model.fit(X_train, y_train, epochs=100, batch_size=5,␣
↪verbose=0)
# evaluate the model
scores = model.evaluate(X_train, y_train)
#print(scores)
print("Neural Network Trainset: \n%s: %.2f%%" %␣
↪(model.metrics_names[1], scores[1]*100))

predictions5 = model.predict(X_test)
#print(predictions)
#print('predictions shape:', predictions.shape)

y_pred5 = (predictions5 > 0.5)
y_pred5 = y_pred5*1 #convert to 0,1 instead of True␣
↪False
cm5 = confusion_matrix(y_test, y_pred5)
print("===================================")
print("===================================")
print("Neural Network on testset confusion matrix")
print(cm5)

## Get accuracy from Confusion matrix
## Position 0,0 and 1,1 are the correct predictions
acc5 = (cm5[0,0] + cm5[1,1]) / sum(sum(cm5))
print("Neural Network on TestSet: Accuracy %.2f%%" %␣
↪(acc5*100))
```

```
[1 7 0]
Regression
================================
[[361  46]
[105 102]]
Regression TrainSet: Accuracy 75.41%
================================
[[82 11]
[30 31]]
Regression Testset: Accuracy 73.38%
================================
================================
================================
Decision Tree
```

(continues on next page)

(continued from previous page)

```
================================
[[407   0]
[  0 207]]
Decsion Tree TrainSet: Accuracy 100.00%
================================
[[68 25]
[32 29]]
Decision Tree Testset: Accuracy 62.99%
================================
================================
================================
Random Forest
================================
[[377  30]
[128  79]]
Random Forest TrainSet: Accuracy 74.27%
================================
[[87  6]
[40 21]]
Random Forest Testset: Accuracy 70.13%
================================
================================
================================
Xgboost
================================
[[389  18]
[ 58 149]]
Xgboost TrainSet: Accuracy 87.62%
================================
Xgboost on testset confusion matrix
[[80 13]
[29 32]]
Xgboost on TestSet: Accuracy 72.73%
================================
20/20 [==============================] - 0s 1ms/step -
↪ loss: 0.5480 - accuracy: 0.7671
Neural Network Trainset:
accuracy: 76.71%
================================
================================
Neural Network on testset confusion matrix
[[81 12]
```

(continues on next page)

(continued from previous page)

```
[29 32]]
Neural Network on TestSet: Accuracy 73.38%
```

## Exercises

We will utilize this dataset [1]:
https://www.kaggle.com/datasets/yasserh/titanic-dataset
The titanic dataset contains the details of passengers aboard the titanic as well as whether they survived the sinking of the titanic. Here are the descriptions of the variables used:

| Variable | Definition | Key |
|---|---|---|
| Survived | Whether the passenger was a survivor | 0 = No, 1 = Yes |
| Pclass | Passenger class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| Name | Passenger name | Text |
| Sex | Sex of passenger | Male or female |
| Age | Age in years | Numerical |
| sibsp | Number of siblings/spouses aboard the Titanic | Numerical |
| Parch | Number of parents/children aboard the Titanic | Numerical |
| Ticket | Ticket number | Text |
| Fare | Passenger fare | Numerical |
| Cabin | Cabin number | Text |
| Embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

In this exercise, try to predict whether or not each of the passengers survived. Here you can put what you have learned into practice. Try to:

- Perform data preparation by cleaning and converting the dataset to a suitable form.
- Create, test, and tune the performance of various classification models.
- Improve the performance by selecting suitable features.

```
import pandas as pd
df = pd.read_csv('Titanic-Dataset.csv')
df.head()
```

```
PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

## References

1. M Yasser H (2021) Titanic dataset. https://www.kaggle.com/datasets/yasserh/titanic-dataset
2. Singh S (2020) Logistic Regression Algorithm Using Data Mining WEKA TOOL. https://medium.com/@singhsiddharth.1998/logistic-regression-algorithm-using-data-mining-weka-tool-d1c88cad7c97. Accessed 27 Apr 2023

# Chapter 4
# Clustering


**Learning Outcomes**

- Understand the difference between supervised and unsupervised algorithms.
- Learn and apply the K-means algorithm for clustering tasks using scikit-learn.
- Learn the Elbow method to select a suitable number of clusters.

## 4.1 Introduction to Clustering

Clustering is the task of dividing the population or data points into a number of groups, such that data points in the same groups are more similar to other data points within the group and dissimilar to the data points in other groups. Clustering is a form of unsupervised algorithm. This means that unlike classification or regression, clustering does not require ground truth labeled data. Such algorithms are capable of finding groups that are not explicitly labeled and identify underlying patterns that might appear in the dataset. In Fig. 4.1, we can see an example of how the data are split into 3 distinct clusters. One of the simplest, yet effective clustering algorithms is the K-means algorithm.

## 4.2 K-means

K-means is used for a variety of cases [1], such as:

- Customer profiling
- Market segmentation
- Computer vision



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_4

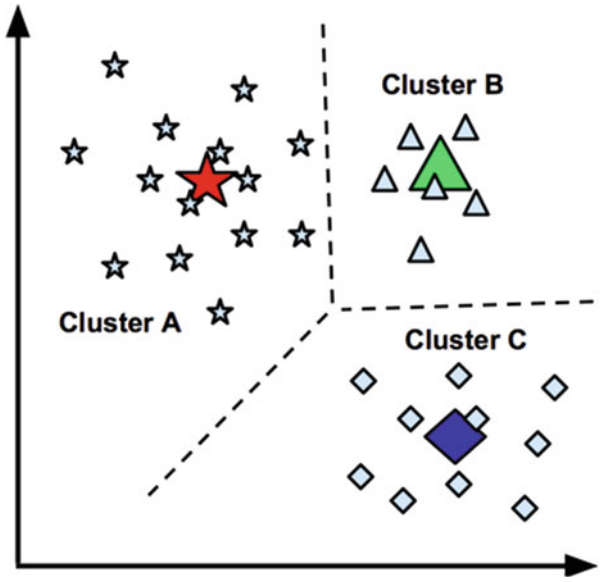



**Fig. 4.1** Example of clustering [1]

- Geo-statistics
- Astronomy

The K-means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. The K-means algorithm aims to choose centroid that minimizes the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||)^2$$

The process for the K-means algorithm is as follows:

1. Ask user how many clusters they would like (e.g., $k = 5$).
2. Randomly guess $k$-cluster center locations.
3. Each data point identifies which center it is closest to according to the sum-of-squares criterion. (Thus each center "owns" a set of data points.)
4. Reposition the $k$-cluster center locations by minimizing the sum-of-squares criterion. This can be achieved by setting the new locations as the average of all the points in a cluster.
5. Repeat steps 3 and 4 until no new data points are added or removed from all clusters, or the pre-defined maximum number of iterations has been reached.

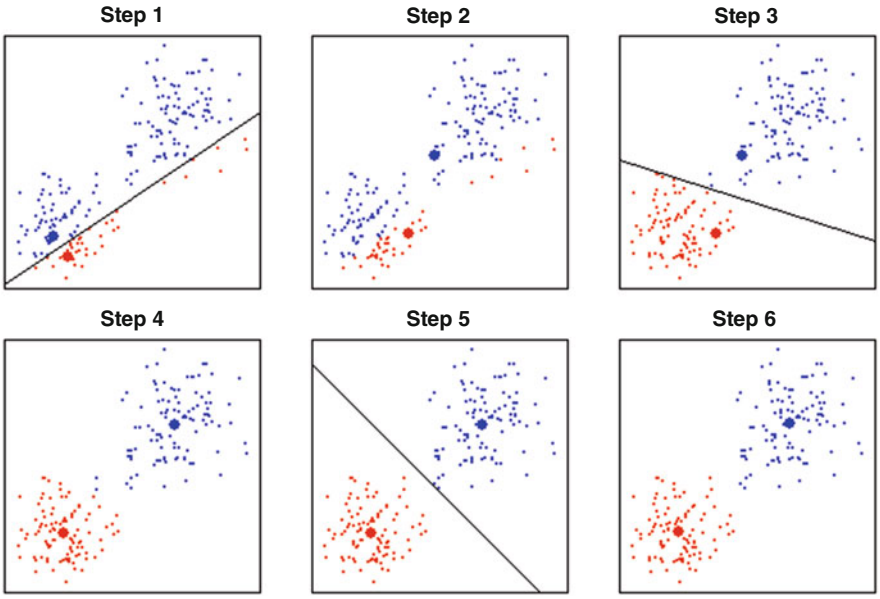



**Fig. 4.2** Step-by-step K-means clustering [1]

Overtime, the associations to each cluster will stabilize and converge to a stable solution. Figure 4.2 shows how the cluster associations progress over each iteration.

## 4.3 The Elbow Method

As you can see in the first step of the K-means algorithm, the user has to specify the number of clusters to be used for the algorithm. We can do this by attempting the K-means for various values of K and visually selecting the K-value using the elbow method as shown in Fig. 4.3. We would like a small sum-of-squares error; however, the sum-of-squares error tends to decrease toward 0 as we increase the value of k. Sum-of-squares will decrease toward 0 with increasing k because when k is equal to the number of data points, each data point is its own cluster, and there will be no error between it and the center of its cluster.

The following code example shows the K-means algorithm and the elbow visualization in figure 4.3 using the Iris dataset that can be obtained from [3]: https://www.kaggle.com/uciml/iris

```
import numpy as np
import pandas as pd
```

(continues on next page)

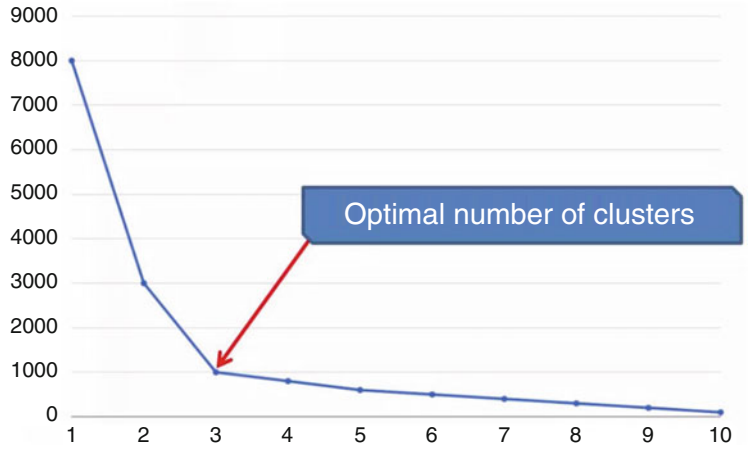



**Fig. 4.3** The elbow method [2]

(continued from previous page)

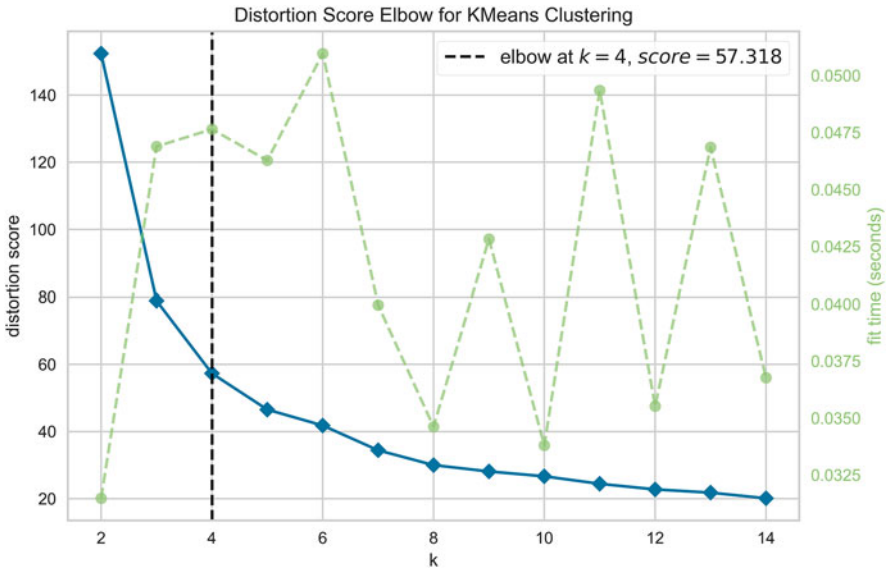
```
df = pd.read_csv("iris.csv")
print(df)
df["Species"].unique()
df = df.replace("Iris-setosa", 0)
df=df.replace("Iris-versicolor", 1)
df = df.replace("Iris-virginica", 2)

X=df.loc[:, ["SepalLengthCm","SepalWidthCm",
↪"PetalLengthCm","PetalWidthCm"]]
Y=df['Species']
print(X)
print(Y)

from sklearn.cluster import KMeans
model=KMeans(n_clusters=3, random_state=2021)
model.fit(X,Y)
pred=model.predict(X)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(pred, Y)
print(cm)

accuracy=(cm[0,0]+cm[1,1]+cm[2,2])/sum(sum(cm))
↪#cm[rows, columns]
print(accuracy)
```

(continues on next page)

(continued from previous page)

```
from yellowbrick.cluster import KElbowVisualizer

visualizer = KElbowVisualizer(model, k=(2,15))

visualizer.fit(X)
visualizer.show()
```

```
     SepalLength  SepalWidth  PetalLength  PetalWidth            Iris
0            5.1         3.5          1.4         0.2     Iris-setosa
1            4.9         3.0          1.4         0.2     Iris-setosa
2            4.7         3.2          1.3         0.2     Iris-setosa
3            4.6         3.1          1.5         0.2     Iris-setosa
4            5.0         3.6          1.4         0.2     Iris-setosa
..           ...         ...          ...         ...             ...
145          6.7         3.0          5.2         2.3  Iris-virginica
146          6.3         2.5          5.0         1.9  Iris-virginica
147          6.5         3.0          5.2         2.0  Iris-virginica
148          6.2         3.4          5.4         2.3  Iris-virginica
149          5.9         3.0          5.1         1.8  Iris-virginica

[150 rows x 5 columns]
     SepalLength  SepalWidth  PetalLength  PetalWidth  Iris
0            5.1         3.5          1.4         0.2     0
1            4.9         3.0          1.4         0.2     0
2            4.7         3.2          1.3         0.2     0
3            4.6         3.1          1.5         0.2     0
4            5.0         3.6          1.4         0.2     0
..           ...         ...          ...         ...   ...
145          6.7         3.0          5.2         2.3     2
146          6.3         2.5          5.0         1.9     2
147          6.5         3.0          5.2         2.0     2
148          6.2         3.4          5.4         2.3     2
149          5.9         3.0          5.1         1.8     2

[150 rows x 5 columns]
[[50  0  0]
 [ 0 48 14]
 [ 0  2 36]]
0.8933333333333333
```




## Exercises

We will utilize this dataset [4]:
https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering

The wine dataset contains the results of a chemical analysis of wines grown in the same region in Italy but may be derived from different cultivars. The analysis determined the quantities of 13 compounds present in the wine varieties. The information of which variety the wine belongs to has been removed to facilitate an unsupervised classification problem.

In this exercise, try to cluster the data into different clusters of wine types.

Here you can put what you have learned into practice. Try to:

- Perform clustering on the dataset using clustering techniques covered in this chapter.
- Select a suitable number of clusters.

```
import pandas as pd
df = pd.read_csv('wine-clustering.csv')
df.head()
```

```
Alcohol  Malic_Acid   Ash  Ash_Alcanity  Magnesium  Total_Phenols  \
0    14.23        1.71  2.43          15.6        127           2.80
1    13.20        1.78  2.14          11.2        100           2.65
2    13.16        2.36  2.67          18.6        101           2.80
```

(continues on next page)

(continued from previous page)

```
3    14.37        1.95  2.50          16.8        113           3.85
4    13.24        2.59  2.87          21.0        118           2.80

Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  Color_Intensity   Hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

OD280  Proline
0   3.92     1065
1   3.40     1050
2   3.17     1185
3   3.45     1480
4   2.93      735
```

## References


1. Madushan D (2017) Introduction to k-means clustering. In: Medium. https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00. Accessed 27 Apr 2023
2. Perera A (2017) Finding the optimal number of clusters for k-means through elbow method using a mathematical approach compared to graphical approach. https://www.linkedin.com/pulse/finding-optimal-number-clusters-k-means-through-elbow-asanka-perera
3. UCIML (2016) Iris species. In: Kaggle. https://www.kaggle.com/uciml/iris. Accessed 27 Apr 2023
4. Wang H (2020) Wine dataset for clustering. https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering

# Chapter 5
# Time Series


**Learning outcomes:**

- Understand what is stationarity and how to test for it.
- Understand concepts such as level, trend, and seasonality.
- Learn to model time series data with simple, double, and triple exponential smoothing.
- Understand difference between autoregressive models and moving average models.
- Learn about SARIMA models.
- Learn about heteroskedasticity and how to predict volatility of time series data.

## 5.1 Introduction to Time Series

A time series is a collection of data points listed in chronological order. Generally, cleaned time series data also ensure that any two consecutive data points have a fixed, defined time interval between them. We perform time series analysis in order to uncover underlying patterns and discover useful insights from the time series data. Time series data analysis is becoming increasingly significant in a variety of industries, including finance, pharmaceuticals, social media, online service providers, research, and many more businesses.

In this chapter, we will cover various forms of time series analysis and demonstrate how to model univariate time series problems.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_5

The dataset we will use will come from the AirPassengers dataset, which can be obtained from this link [6]:
https://www.kaggle.com/datasets/rakannimer/air-passengers

```
import pandas as pd
df = pd.read_csv("AirPassengers.csv")
```

```
df.head()
```

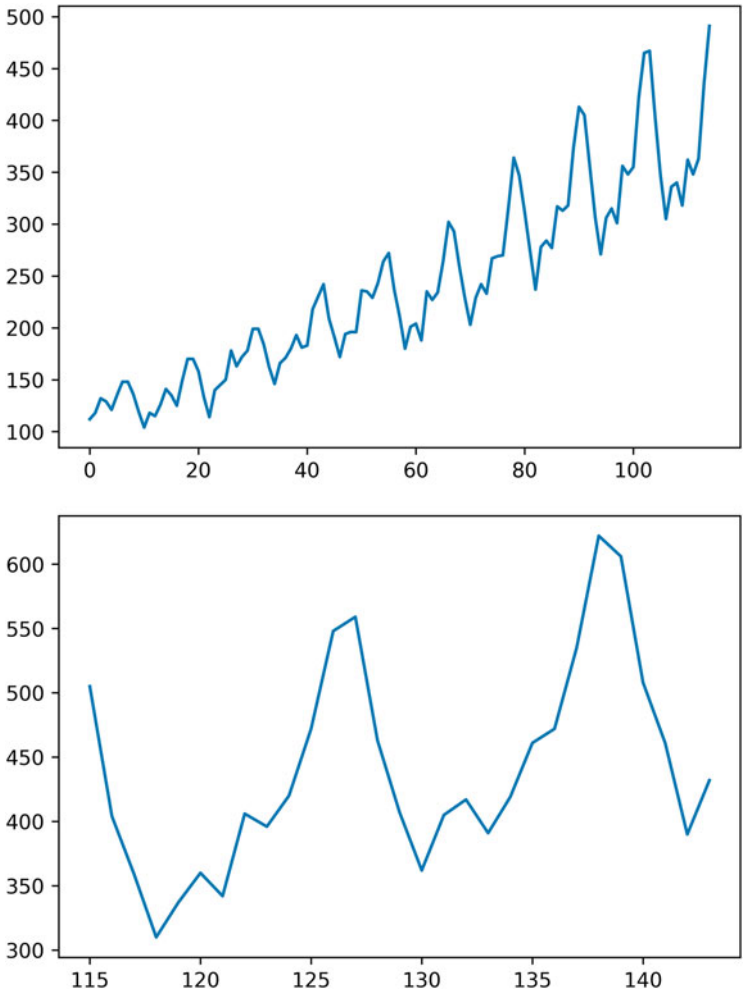
```
     Month  #Passengers
0  1949-01          112
1  1949-02          118
2  1949-03          132
3  1949-04          129
4  1949-05          121
```

```
df = df.drop(['Month'], axis=1)
df.iloc[:,0]=pd.to_numeric(df.iloc[:,0], errors=
↪"coerce")
```

```
df = df.dropna()
df = df.interpolate()
column_name=df.columns[0]
```

```
n_row=len(df)
train_row = int(0.8 * n_row)
train = df[0:train_row]
test = df[train_row:]
pred = test.copy()
```

```
train.plot()
test.plot()
```




## 5.2 Stationarity

There are various types of time series trends as shown in Fig. 5.1. One particular time series to take note of are stationary time series. A time series is only considered stationary when its mean, variance, and autocovariance do not change across time.

If the time series is not stationary, you will need to use a model that captures trend and/or seasonality.

After developing the model, you can remove the trend and seasonality component away from the time series and then the label the residuals as stationary. If the residuals left are still not stationary, then the model used is not suitable or the time series cannot be accurately predicted.

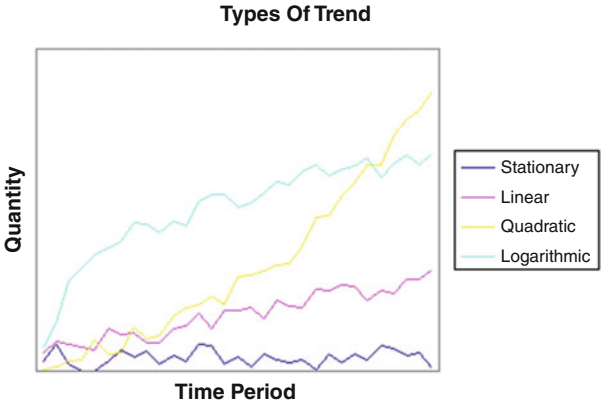



**Fig. 5.1** Types of trends [5]

We can test for stationarity using the Dickey–Fuller test if >0.05 => non stationary => needs to use trend and seasonality.

```
from statsmodels.tsa.stattools import adfuller

dicky = adfuller(df)
print(dicky[1])
```

```
0.9918802434376408
```

In this case, the data have a p-value >0.05, and hence, it is not stationary.

## 5.3 Level, Trend, and Seasonality

Level: The average value of the series.
Trend: The increasing or decreasing value in the series.
Seasonality: The repeating short-term cycle in the series.
Noise: The random variation in the series.

Both trends and seasonality can be additive or multiplicative as shown in Fig. 5.2.

We can decompose a time series into its various components as follows using an additive model.

```
from statsmodels.tsa.seasonal import seasonal_
↪decompose
import matplotlib.pyplot as plt
```

(continues on next page)

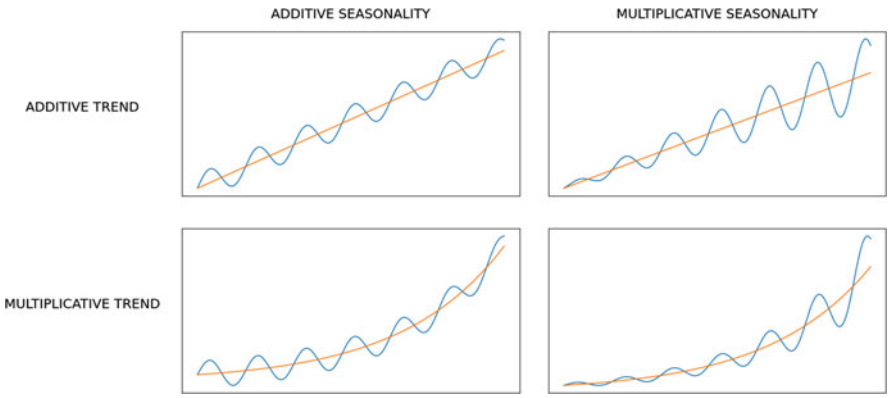
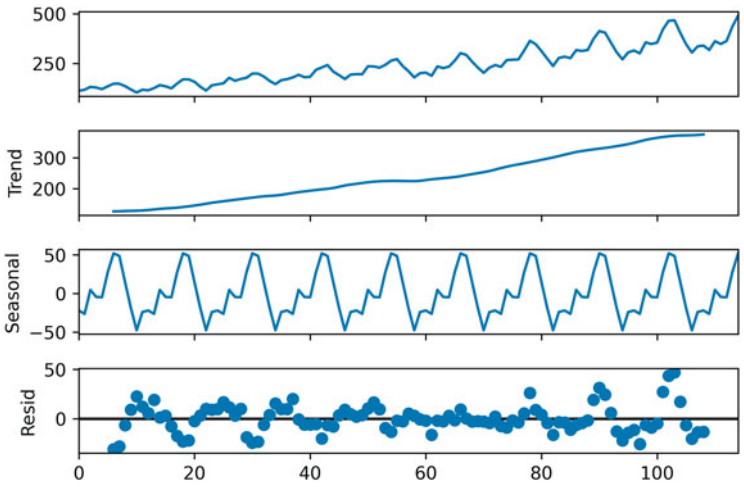



**Fig. 5.2** Types of trends [5]

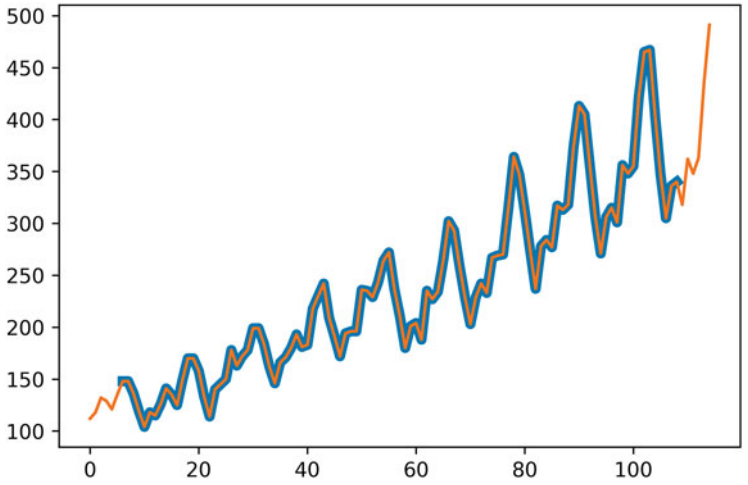
(continued from previous page)

```
decomposition = seasonal_decompose(train, model=
↪'additive', period=12)
trend = decomposition.trend.dropna()
seasonal=decomposition.seasonal.dropna()
residual=decomposition.resid.dropna()

decomposition.plot()
plt.show()
```



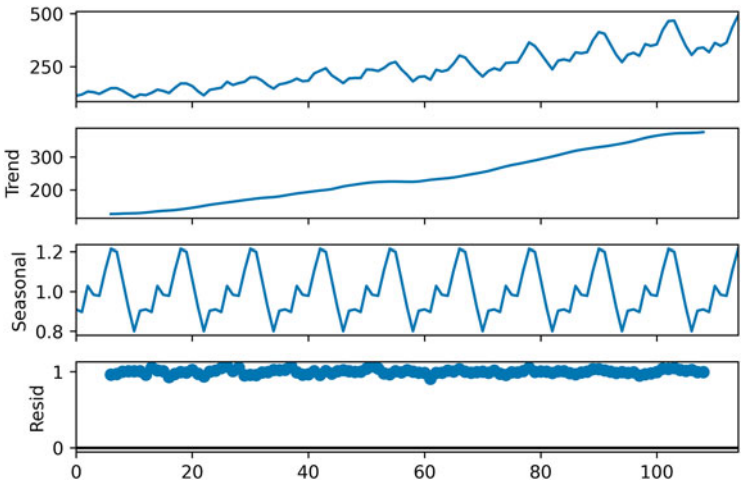
We can recompose the time series back by adding the various components together in an additive model.

```
series =  trend + seasonal + residual
plt.plot(series, linewidth=5)
plt.plot(train)
plt.show()
```



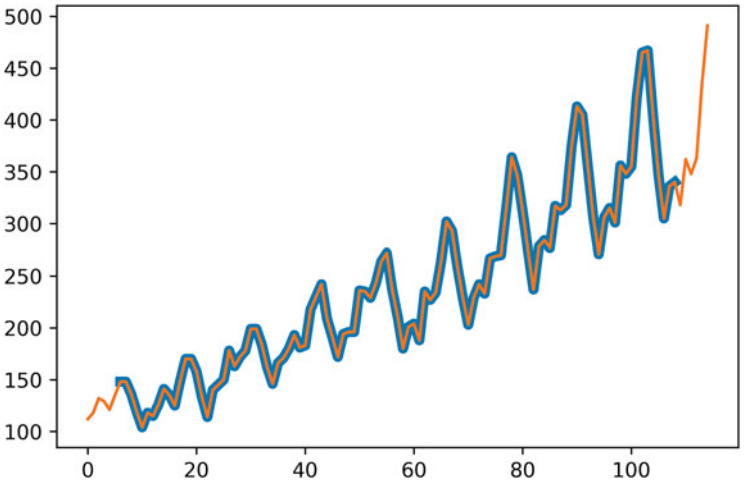

Similarly, we can also decompose a time series using a multiplicative model.

```
decomposition = seasonal_decompose(train, model=
↪'multiplicative', period=12)
trend = decomposition.trend.dropna()
seasonal=decomposition.seasonal.dropna()
residual=decomposition.resid.dropna()

decomposition.plot()
plt.show()
```




And recompose it back by multiplying the elements.

```
series =  trend*seasonal*residual
plt.plot(series, linewidth=5)
plt.plot(train)
plt.show()
```



## 5.4 Exponential Smoothing

### *5.4.1 Simple Exponential Smoothing*

Simple exponential smoothing forecasts future values based on a weighted average of previous values [1]. It is suitable for time series that have no trends and no seasonality.

The model only contains a level component.

$$\hat{y}_t = \alpha \cdot y_{t-1} + (1 - \alpha) \cdot \hat{y}_{t-1}$$

The current level is equivalent to a mixture of the previous level and the currently observed value at time $t$. Alpha is the weight placed on the current observation and is called a smoothing constant.

When Alpha is equal to 1, the predicted value is only dependent on the most recent observed value.

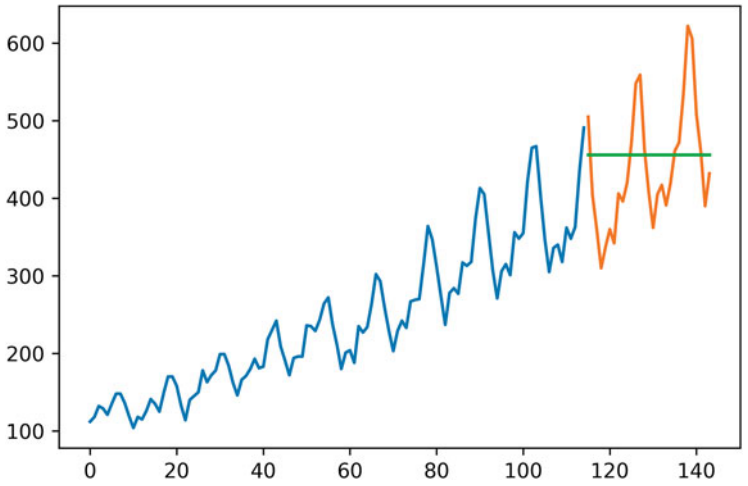
When Alpha is equal to 0, the predicted value is only dependent on the past value where l0 = x0.

Using simple exponential smoothing, let us forecast the entire test period.

```
from statsmodels.tsa.api import SimpleExpSmoothing,␣
↪Holt, ExponentialSmoothing
import numpy as np
from sklearn.metrics import mean_squared_error

model = SimpleExpSmoothing(np.asarray(train.iloc[:,
↪0])).fit(smoothing_level=0.6)
#0.6 is for y(t-1) and 0.4 is for y(t-1 and below)
pred['ES'] = model.forecast(len(test))

mean_squared_error(pred['ES'],test[column_name])**0.5
```

```
79.71276692933388
```

```
plt.plot(train)
plt.plot(test)
plt.plot(pred['ES'])
plt.show()
```




### 5.4.2 *Double Exponential Smoothing (Holt's Exponential Smoothing)*

Double exponential smoothing forecasts future values based on the level and a trend [1]. It is suitable for time series that have only trends and no seasonality.

The model only contains a level component and a trend component. A trend can be either additive or multiplicative, and the following shows the formulas for an additive model.

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)(b_{t-1})$$
$$\hat{y}_{t+1} = \ell_t + b_t$$

The trend is equivalent to a mixture of the previous trend value and the change in the level at time $t$. Beta is the weight placed on the change in the level and is another smoothing constant.

Note that the equation for level has been adapted to include the trend forecast at $t-1$.

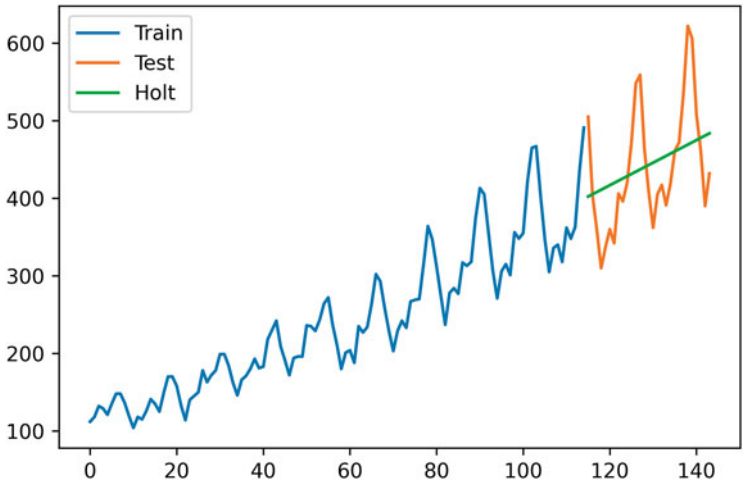
When Beta is equal to 1, the predicted value is only dependent on the most recent difference in level values.

When Beta is equal to 0, the predicted value is only dependent on the past trends.

```
# exponential = False for additive model and True for␣
↪multiplicative model
model = Holt(np.asarray(train.iloc[:,0]),␣
↪exponential=False).fit(smoothing_level = 0.1,␣
↪smoothing_trend = 0.1)
pred['Holt'] = model.forecast(len(test))
mean_squared_error(pred['Holt'],test[column_name])**0.
↪5
```

```
70.67897411947663
```

```
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.plot(pred['Holt'], label='Holt')
plt.legend(loc='best')
plt.show()
```



### 5.4.3 *Triple Exponential Smoothing (Holt–Winters Exponential Smoothing)*

Triple exponential smoothing forecasts future values based on the level, a trend, and seasonality [1]. It is suitable for time series that have trends and seasonality.

The model only contains a level component, a trend component, and a seasonality component. Seasonality can be either additive or multiplicative, and the following shows the formulas for an additive model.

$$\ell_t = \alpha(y_t - s_{t-p}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)(b_{t-1})$$
$$s_t = \gamma(y_t - \ell_t) + (1-\gamma)s_{t-p}$$
$$\hat{y}_{t+m} = \ell_t + mb_t + s_{t-p+1+(m-1)modp}$$

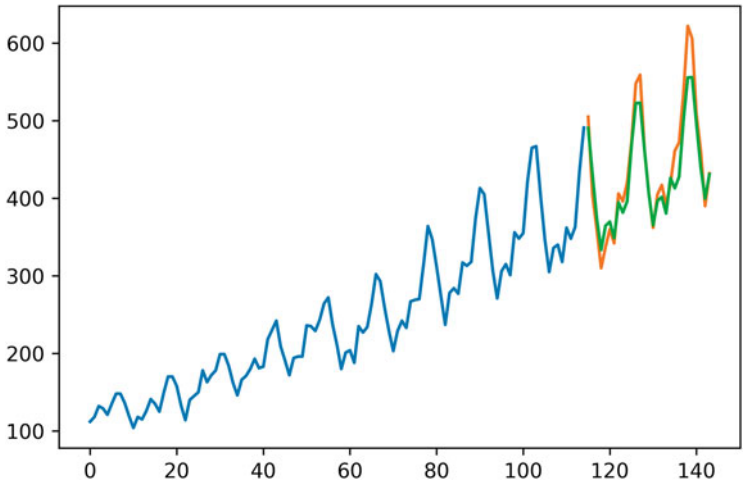
The seasonality is equivalent to a mixture of the previous seasonality value and the difference between the current level and observation values at time t. Gamma is the weight placed on the change in the seasonality and is another smoothing constant. p is the seasonality period.

When Gamma is equal to 1, the predicted value is only dependent on the most recent difference between the current observation and level values.
When Gamma is equal to 0, the predicted value is only dependent on the past seasonality.

```
#Parameter https://www.statsmodels.org/dev/generated/
↪statsmodels.tsa.holtwinters.ExponentialSmoothing.
↪html

model = ExponentialSmoothing(train, trend='add',␣
↪seasonal='mul', seasonal_periods=12).fit()
pred['Holt_Winter'] = model.forecast(len(test))
pred = pred.dropna()
mean_squared_error(pred['Holt_Winter'],pred[column_
↪name])**0.5
```

```
26.803585376637876
```

```
plt.plot(train)
plt.plot(test)
plt.plot(pred['Holt_Winter'])
plt.show()
```




## 5.5 Moving Average Smoothing

Moving average smoothing is a simple and effective time series forecasting technique that may be used in feature engineering or prediction [2].

It aims to eliminate noise and better reveal the signal of underlying causal processes by reducing fine-grained variance between time steps in time series. This is done by calculating a new series comprised of the average of raw observations within a sliding window from the original time series. Thus, the moving average requires the window size to be specified.

The term "moving average" refers to how the window of values indicated by the window size is moving along the time series to determine the values in the new series. Simple moving average, weighted moving average, and exponential moving average are some of the most commonly used moving average smoothing techniques used in time series forecasting.

Simple Moving Average: $SMA_t = \frac{X_1+X_2+....+X_n}{n}$

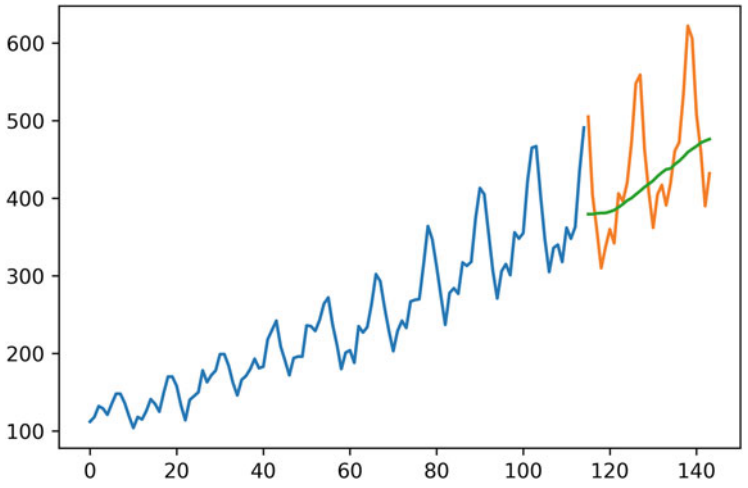
Weighted Moving Average: $WMA_t = (W_1X_1 + W_2X_2 + .... + W_nX_n)$

Exponential Moving Average: $EMA_t = EMA_{t-1} + \alpha(X_t - EMA_{t-1})$

```
pred['SMA'] = df.iloc[:,0].rolling(12).mean()

mean_squared_error(pred['SMA'],test[column_name])**0.5
```

```
71.81844893395912
```

```
plt.plot(train)
plt.plot(test)
plt.plot(pred['SMA'])
plt.show()
```




## 5.6 Autoregression

Autoregression means regression on self as we use data from the same input variable at earlier time steps [3].

Regression models, such as linear regression, predict an output value from a linear combination of input values.

$y = b_0 + b_1 * X_1$

where $y$ is the prediction, $b_0$ and $b_1$ are coefficients learned from the training data, and $X$ is an input value.

This approach may be applied to time series in which the input variables are taken as observations at past time steps, which are called lag variables. This is called autoregression.

For example, we may estimate the value at time step t using the observations from the previous two time steps ($t-1$ and $t-2$).

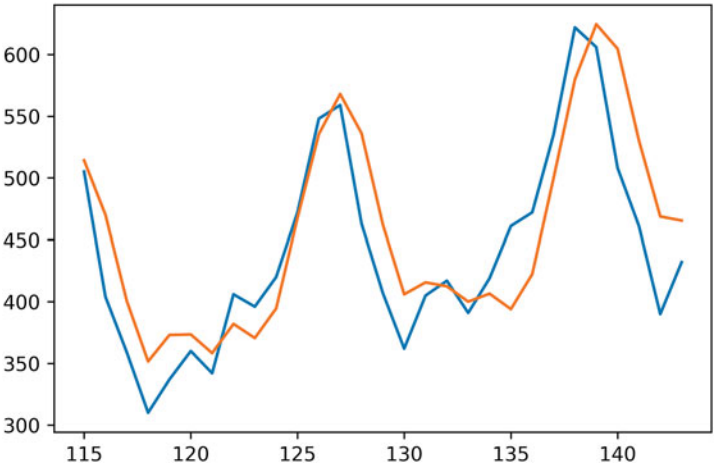
$X_t = b_0 + b_1 * X_{t-1} + b_2 * X_{t-2}$

```
from matplotlib import pyplot
from statsmodels.tsa.ar_model import AutoReg
```

(continues on next page)

(continued from previous page)

```
from math import sqrt

model = AutoReg(train, lags=12)
model = model.fit()

predictions = model.predict(start=min(test.index), end=max(test.index), dynamic=False)
predictions = predictions[predictions.index.isin(test.index)]

# plot results
plt.plot(test)
plt.plot(predictions)
plt.show()

rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
```




```
Test RMSE: 43.255
```

## 5.7 Moving Average Process

The moving average model here refers to a moving average process, which is different from moving average smoothing. A moving average model is used to anticipate future values, whereas moving average smoothing is used to estimate

previous value trend cycles [4]. The moving average model uses errors of previous forecasts to estimate future errors.

$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q}$

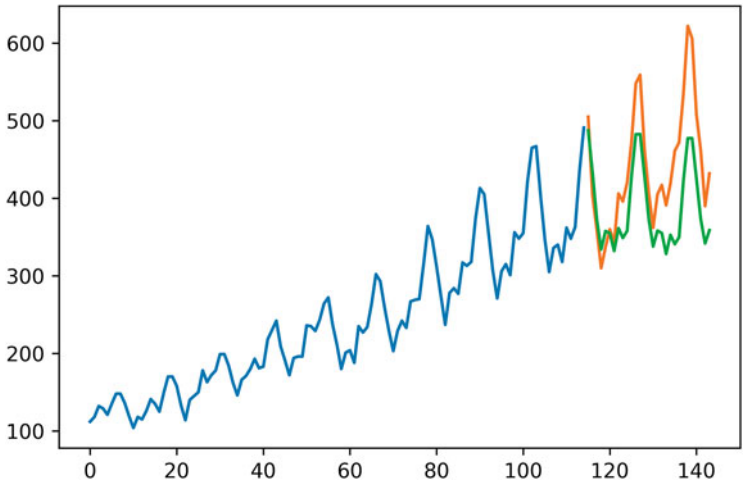
where $\varepsilon_t$ is assumed to be white noise.

A moving average model of order q is called a MA(q) model. As you can see from the formula above, each value of $y_t$ can be viewed as a weighted moving average of the past few forecast errors.

```
# ARMA example
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(train.iloc[:,0], order=(12, 0, 12)).
↪fit()

pred['ARMA'] = model.forecast(max(test.index))
mean_squared_error(pred['ARMA'], pred[column_
↪name])**0.5
```

```
69.89049777416116
```

```
plt.plot(train)
plt.plot(test)
plt.plot(pred['ARMA'])
plt.show()
```



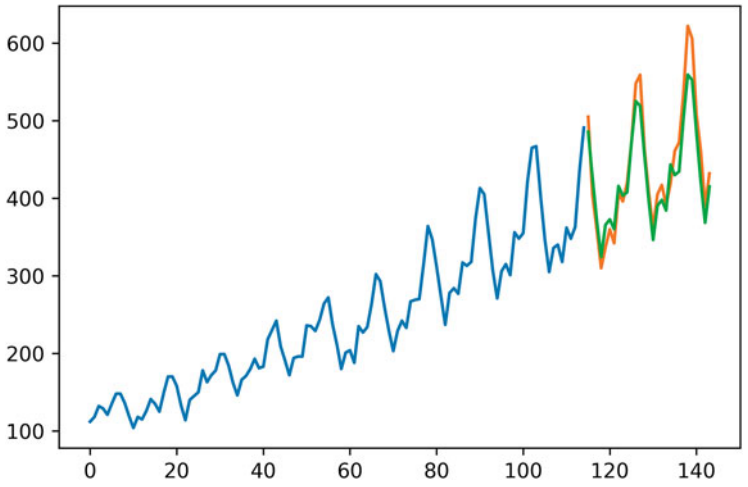
## 5.8 SARIMA

SARIMA is short for seasonal autoregressive integrated moving average model. This model extends the ARIMA algorithm by adding a seasonality component. Thus, SARIMA combines seasonality, autoregression, differencing, and moving average process and has many parameters to tune.
An ARIMA model is characterized by 3 terms: p, d, q,
where
p is the order of the AR term,
q is the order of the MA term,
and d is the number of differencing required to make the time series stationary.
Seasonal Elements
There are four seasonal elements that are not part of ARIMA that must be configured; they are:
P: Seasonal autoregressive order.
D: Seasonal difference order.
Q: Seasonal moving average order.
m: The number of time steps for a single seasonal period.

```
import statsmodels.api as sm

model = sm.tsa.statespace.SARIMAX(train.iloc[:,0],␣
↪order=(2, 2, 2), seasonal_order=(1,1,1,12)).fit()

pred['SARIMA'] = model.forecast(max(test.index))
mean_squared_error(pred['SARIMA'],pred[column_
↪name])**0.5
```

```
25.61952170892105
```

```
plt.plot(train)
plt.plot(test)
plt.plot(pred['SARIMA'])
plt.show()
```




```
# check dicky fuller on the residual

result = adfuller(residual)
print('p-value: %f' % result[1])

# p vlaue > 0.05 means there non-stationary, need to␣
↪use non stationary model
```

```
p-value: 0.000004
```

## 5.9 ARCH/GARCH

Heteroskedasticity refers to a situation where the standard deviations of the residuals in a regression model are non-constant over different values of an independent variable or time periods.

Autoregressive conditional heteroskedasticity (ARCH) is a statistical model used to analyze volatility in time series in order to forecast future volatility.

GARCH is appropriate for time series data where the variance of the error term is serially autocorrelated following an autoregressive moving average process.

The innovation is the difference between the observed value of a variable at time t and the optimal forecast of that value based on information available prior to time $t$.
p: The number of lag variances.
q: The number of lag residual errors.

```
from arch import arch_model
model = arch_model(df, vol="ARCH", p=1)
results = model.fit()
results.summary()
```

```
Iteration:      1,   Func. Count:      5,   Neg. LLF: 854.6007658170145
Iteration:      2,   Func. Count:     11,   Neg. LLF: 854.5884325691222
Iteration:      3,   Func. Count:     16,   Neg. LLF: 854.2541165562286
Iteration:      4,   Func. Count:     21,   Neg. LLF: 852.6533203012712
Iteration:      5,   Func. Count:     26,   Neg. LLF: 846.5727355761943
Iteration:      6,   Func. Count:     32,   Neg. LLF: 845.7983348298947
Iteration:      7,   Func. Count:     37,   Neg. LLF: 845.7954504118902
Iteration:      8,   Func. Count:     42,   Neg. LLF: 845.7953853667937
Iteration:      9,   Func. Count:     47,   Neg. LLF: 845.7953355810648
Iteration:     10,   Func. Count:     52,   Neg. LLF: 845.7951657021088
Iteration:     11,   Func. Count:     57,   Neg. LLF: 845.7947852612426
Iteration:     12,   Func. Count:     62,   Neg. LLF: 845.7936868695066
Iteration:     13,   Func. Count:     67,   Neg. LLF: 845.790907784166
Iteration:     14,   Func. Count:     72,   Neg. LLF: 845.7834353508538
Iteration:     15,   Func. Count:     77,   Neg. LLF: 845.7637034098763
Iteration:     16,   Func. Count:     82,   Neg. LLF: 845.7110763686674
Iteration:     17,   Func. Count:     87,   Neg. LLF: 845.5732564306693
Iteration:     18,   Func. Count:     92,   Neg. LLF: 845.1900634849089
Iteration:     19,   Func. Count:     97,   Neg. LLF: 844.2688848009523
Iteration:     20,   Func. Count:    102,   Neg. LLF: 843.464456336123
Iteration:     21,   Func. Count:    107,   Neg. LLF: 842.3234724262779
Iteration:     22,   Func. Count:    112,   Neg. LLF: 842.0273124952928
Iteration:     23,   Func. Count:    117,   Neg. LLF: 841.9980327602752
Iteration:     24,   Func. Count:    122,   Neg. LLF: 841.2787441430107
Iteration:     25,   Func. Count:    129,   Neg. LLF: 840.8270304418386
Iteration:     26,   Func. Count:    135,   Neg. LLF: 840.6654988236875
Iteration:     27,   Func. Count:    140,   Neg. LLF: 840.5572235842828
Iteration:     28,   Func. Count:    145,   Neg. LLF: 840.5342860613555
Iteration:     29,   Func. Count:    150,   Neg. LLF: 840.5335370026474
Iteration:     30,   Func. Count:    155,   Neg. LLF: 840.5335343507581
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 840.5335343695717
            Iterations: 30
            Function evaluations: 159
            Gradient evaluations: 30
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - ARCH Model Results
==============================================================================
Dep. Variable:            #Passengers   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                       ARCH   Log-Likelihood:               -840.534
Distribution:                  Normal   AIC:                           1687.07
Method:            Maximum Likelihood   BIC:                           1695.98
                                        No. Observations:                  144
Date:                Tue, Feb 21 2023   Df Residuals:                      143
Time:                        23:23:21   Df Model:                            1
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu           202.4031      8.359     24.213 1.624e-129 [1.860e+02,2.188e+02]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
omega         343.7569     104.579      3.287  1.012e-03 [1.388e+02,5.487e+02]
alpha[1]        1.0000  5.702e-02     17.538  7.367e-69      [  0.888,  1.112]
============================================================================

Covariance estimator: robust
"""
```

```
from arch import arch_model
model = arch_model(df, vol="GARCH", p=1,q=1)
results = model.fit()
results.summary()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 867.8646906481998
Iteration:      2,   Func. Count:     12,   Neg. LLF: 858.811426534049
Iteration:      3,   Func. Count:     19,   Neg. LLF: 857.6407923629904
Iteration:      4,   Func. Count:     25,   Neg. LLF: 857.5452375347352
Iteration:      5,   Func. Count:     31,   Neg. LLF: 857.0520810189686
Iteration:      6,   Func. Count:     37,   Neg. LLF: 854.7269894558582
Iteration:      7,   Func. Count:     44,   Neg. LLF: 853.324769527305
Iteration:      8,   Func. Count:     51,   Neg. LLF: 846.8532975963085
Iteration:      9,   Func. Count:     59,   Neg. LLF: 843.2081443338757
Iteration:     10,   Func. Count:     65,   Neg. LLF: 840.9208681923458
Iteration:     11,   Func. Count:     72,   Neg. LLF: 840.812365925325
Iteration:     12,   Func. Count:     78,   Neg. LLF: 840.6235710976691
Iteration:     13,   Func. Count:     84,   Neg. LLF: 840.6226903990031
Iteration:     14,   Func. Count:     90,   Neg. LLF: 840.6225148219894
Iteration:     15,   Func. Count:     96,   Neg. LLF: 840.6225006840967
Iteration:     16,   Func. Count:    102,   Neg. LLF: 840.6224171929287
Iteration:     17,   Func. Count:    108,   Neg. LLF: 840.6219978934255
Iteration:     18,   Func. Count:    114,   Neg. LLF: 840.6199192651045
Iteration:     19,   Func. Count:    120,   Neg. LLF: 840.6100925694149
Iteration:     20,   Func. Count:    126,   Neg. LLF: 840.5729297977884
Iteration:     21,   Func. Count:    132,   Neg. LLF: 840.5351083598802
Iteration:     22,   Func. Count:    138,   Neg. LLF: 840.5335777331641
Iteration:     23,   Func. Count:    144,   Neg. LLF: 840.533535251781
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 840.5335346045151
            Iterations: 23
            Function evaluations: 145
            Gradient evaluations: 23
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:            #Passengers   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -840.534
Distribution:                  Normal   AIC:                           1689.07
Method:            Maximum Likelihood   BIC:                           1700.95
                                        No. Observations:                  144
Date:                Tue, Feb 21 2023   Df Residuals:                      143
Time:                        23:23:22   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu           202.4018      9.040     22.389 5.019e-111 [1.847e+02,2.201e+02]
                          Volatility Model
============================================================================
```

(continues on next page)

(continued from previous page)

```
                 coef    std err          t      P>|t|    95.0% Conf. Int.
-----------------------------------------------------------------------------
omega        343.8344    135.831      2.531  1.136e-02 [ 77.611,6.101e+02]
alpha[1]       1.0000      0.260      3.842  1.222e-04   [  0.490,  1.510]
beta[1]    5.3178e-17      0.261  2.040e-16      1.000   [ -0.511,  0.511]
=============================================================================

Covariance estimator: robust
"""
```

## Exercises

We will utilize this dataset [7]:
https://www.kaggle.com/datasets/bhaveshsonagra/monthly-milk-production

The time series dataset measures pounds of milk produced per cow as its unit per month from January 1962 to December 1975.

In this exercise, try to forecast the production of milk using time series techniques.

Here you can put what you have learned into practice. Try to:

- Test for stationarity.
- Experiment with moving average smoothing.
- Create, test, and tune the performance of various time series forecasting techniques covered in this chapter.

```
import pandas as pd
df = pd.read_csv('monthly-milk-production-pounds.csv')
df.head()
```

```
Month  Monthly milk production: pounds per cow. Jan 62 ? Dec 75
0  1962-01                                                589.0
1  1962-02                                                561.0
2  1962-03                                                640.0
3  1962-04                                                656.0
4  1962-05                                                727.0
```

## References

1. Baysan (2022) Introduction to time series forecasting: smoothing methods. https://medium.com/codex/introduction-to-time-series-forecasting-smoothing-methods-9a904c00d0fd. Accessed 27 Apr 2023
2. Brownlee J (2016) Moving average smoothing for data preparation and time series forecasting in Python. https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/. Accessed 27 Apr 2023

3. Brownlee J (2017) Autoregression models for time series forecasting with Python. https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python. Accessed 27 Apr 2023
4. Hyndman RJ, Athanasopoulos G (2021) Forecasting: principles and practice. OTexts
5. iPredict (2004) iPredict—time-series forecasting software. In: iPredict. http://www.ipredict.it/TimeSeriesForecasting/. Accessed 27 Apr 2023
6. Rakannimer (2017) Air passengers. In: Kaggle. https://www.kaggle.com/datasets/rakannimer/air-passengers. Accessed 27 Apr 2023
7. Sonagra B (2020) Monthly milk production. https://www.kaggle.com//datasets/bhaveshsonagra/monthly-milk-production

# Chapter 6
# Convolutional Neural Networks


**Learning outcomes:**

- Understand how convolution, pooling, and flattening operations are performed.
- Perform an image classification task using convolutional neural networks.
- Familiarize with notable convolutional neural network (CNN) architectures.
- Understand transfer learning and finetuning.
- Perform an image classification task through finetuning a convolutional neural network previously trained on a separate task.
- Exposure to various applications of convolutional neural networks.

A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layers. The main problem with fully connected neural networks is that the number of weights required is very large for certain types of data. For example, an image of $224 \times 224 \times 3$ would require 150528 weights in just the first hidden layer and will grow quickly for even bigger images. You can imagine how computationally intensive things would become once the images reach dimensions as large as 8K resolution images ($7680 \times 4320$), and training such a network would require a lot of time and resources.

However for image data, repeating patterns can occur in different places. Hence, we can train many smaller detectors, capable of sliding across an image, to take advantage of the repeating patterns as shown in Fig. 6.1. This would reduce the number of weights required to be trained.

A convolutional neural network is a neural network with some convolution layers (and some other layers). A convolution layer has a number of filters that do the



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_6

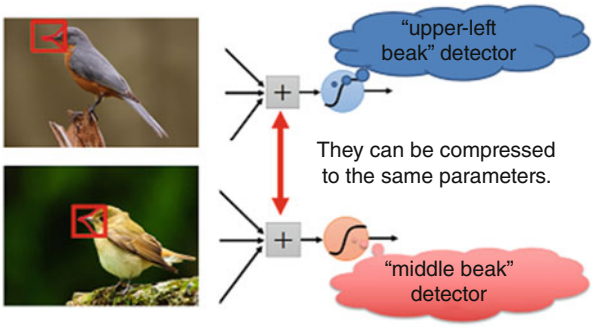



**Fig. 6.1** Illustration of sliding window detectors [4]

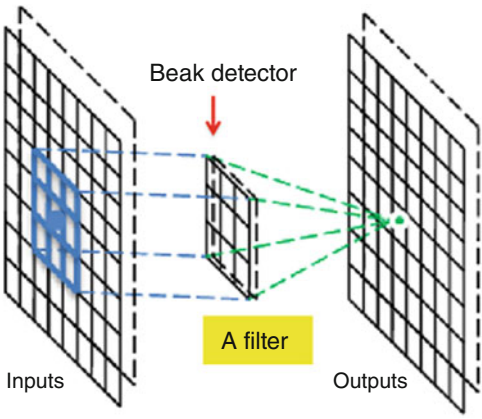



**Fig. 6.2** Illustration of convolution operation [4]

convolution operation. Figure 6.2 shows how the input image is processed with a convolution filter to produce the output feature maps.

## 6.1 The Convolution Operation

The convolution operation as shown in Fig. 6.3 is very similar to image processing filters such as the Sobel filter and Gaussian filter. The kernel slides across an image and multiplies the weights with each aligned pixel, element-wise across the filter. Afterward, the bias value is added to the output.

There are three hyperparameters deciding the spatial of the output feature map:

- Stride (S) is the step each time we slide the filter. When the stride is 1, then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice), then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.
- Padding (P): The inputs will be padded with a border of size according to the value specified. Most commonly, zero-padding is used to pad these locations. In

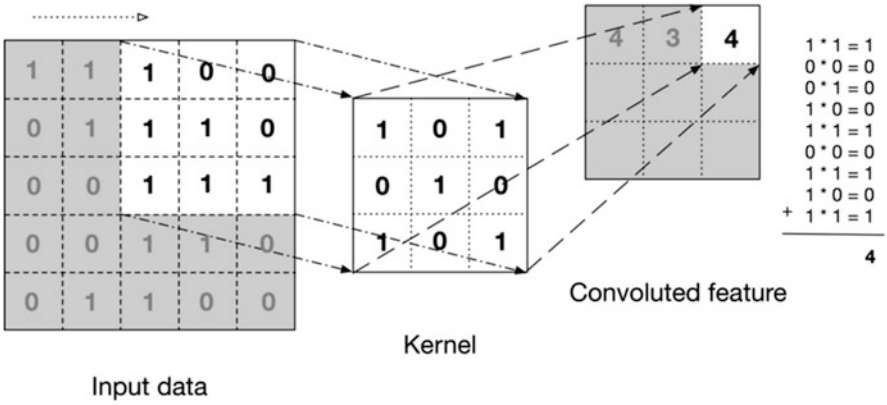



**Fig. 6.3** Illustration of convolution operation for single channel [6]

neural network frameworks (Caffe, TensorFlow, Pytorch, MXNet), the size of this zero-padding is a hyperparameter. The size of zero-padding can also be used to control the spatial size of the output volumes.

- Depth (D): The depth of the output volume is a hyperparameter too, and it corresponds to the number of filters we use for a convolution layer.

Given W as the width of input, and F is the width of the filter, with P and S as padding, the output width will be: $(W + 2P - F)/S + 1$ Generally, set $P = (F - 1)/2$ when the stride is S=1 ensures that the input volume and output volume will have the same size spatially.

For an input of $7 \times 7 \times 3$ and an output depth of 2, we will have 6 kernels as shown below. 3 for the first depth output and another 3 for the second depth output. The outputs of each filter are summed up to generate the output feature map.

In the example shown in Fig. 6.4, the output from each Kernel of Filter W1 is as follows:

Output of Kernel 1 = 1
Output of Kernel 2 = −2
Output of Kernel 3 = 2
Output of Filter W1 = Output of Kernel 1 + Output of Kernel 2 + Output of Kernel 3 + bias = 1−2 + 2 + 0 = 1

## 6.2 Pooling

Nowadays, a CNN always exploits extensive weight-sharing to reduce the degrees of the freedom of models. A pooling layer helps reduce computation time and gradually build up spatial and configuration invariance. For image understanding, pooling layer helps extract more semantic meaning. The max pooling layer simply

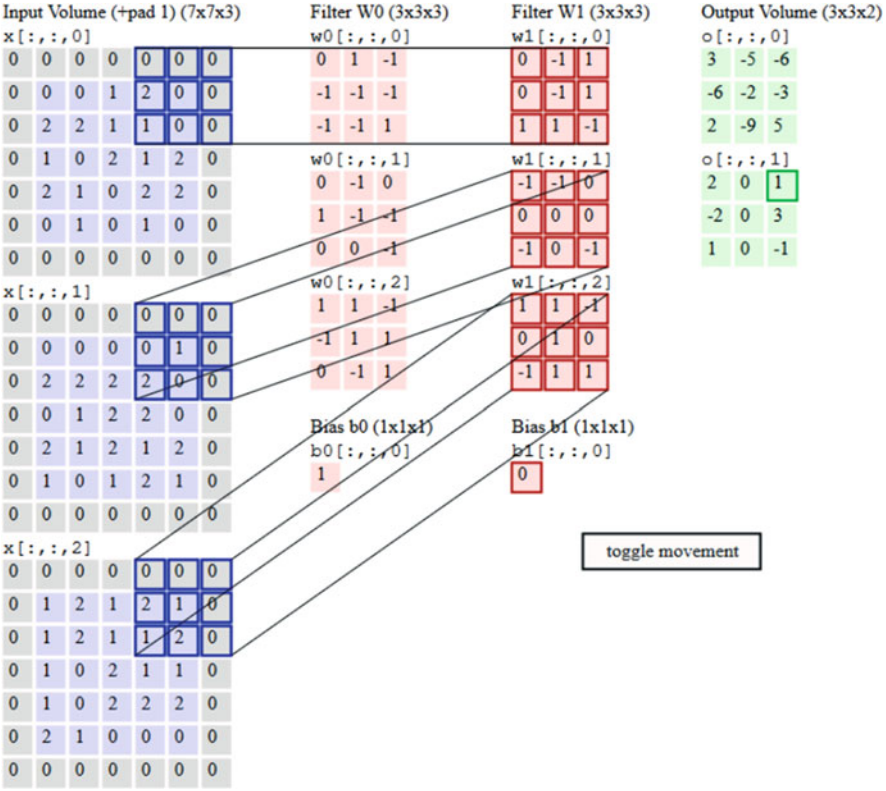



**Fig. 6.4** Example of convolution operation for multiple channels [10]

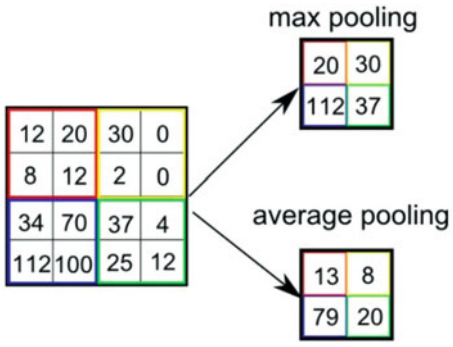



**Fig. 6.5** Example of max pooling and average pooling [6]

returns the maximum value over the values that the kernel operation is applied on. The example below in Fig. 6.5 illustrates the outputs of a max pooling and average pooling operation, respectively, given a kernel of size 2 and stride 2.

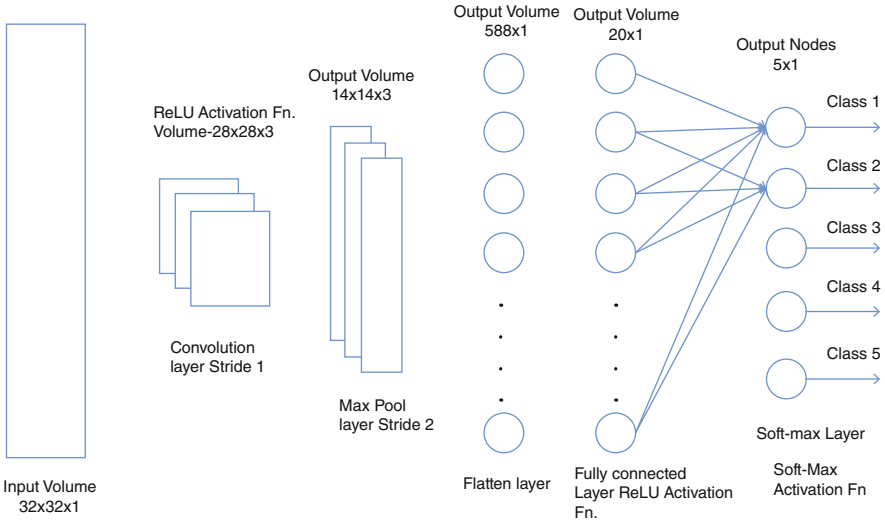



**Fig. 6.6** Example of CNN using flattening for classification [1]

## 6.3 Flattening

Adding a fully connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolution layer. The fully connected layer is learning a possibly non-linear function in that space.

As shown in Fig. 6.6, flattening the image into a column vector allows us to convert our input image into a suitable form for our multi-level perceptron. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax classification technique.

## 6.4 Building a CNN

We will build a small CNN using convolution layers, max pooling layers, and dropout layers in order to predict the type of fruit in a picture.

The dataset we will use is the fruits 360 dataset. You can obtain the dataset from this link [5]:
https://www.kaggle.com/moltean/fruits

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O␣
↪(e.g. pd.read_csv)
import os

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,␣
↪Flatten
from tensorflow.keras.layers import Conv2D,␣
↪MaxPooling2D
from tensorflow.keras import optimizers
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.image import␣
↪ImageDataGenerator

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pathlib
```

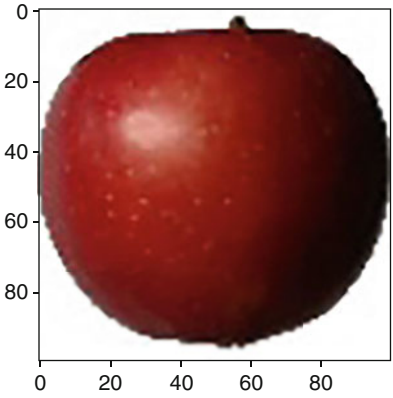
```
train_root  =pathlib.Path("D:/Programming Stuff/Teoh
↪'s Slides/book-ai-potato (docs)/fruits-360_dataset/
↪fruits-360/Training")
test_root = pathlib.Path("D:/Programming Stuff/Teoh's␣
↪Slides/book-ai-potato (docs)/fruits-360_dataset/
↪fruits-360/Test")
```

```
batch_size = 10
```

```
from skimage import io
image = io.imread("D:/Programming Stuff/Teoh's Slides/
↪/book-ai-potato (docs)/fruits-360_dataset/fruits-
↪360/Training/Apple Braeburn/101_100.jpg")
print(image.shape)
io.imshow(image)
```

```
(100, 100, 3)
```

```
<matplotlib.image.AxesImage at 0x1543232f070>
```




```
Generator = ImageDataGenerator()
train_data = Generator.flow_from_directory(train_root,
 ↪ (100, 100), batch_size=batch_size)
test_data = Generator.flow_from_directory(test_root,␣
 ↪(100, 100), batch_size=batch_size)
```

```
Found 67692 images belonging to 131 classes.
Found 22688 images belonging to 131 classes.
```

```
num_classes = len([i for i in os.listdir(train_root)])
print(num_classes)
```

```
131
```

```
model = Sequential()

model.add(Conv2D(16, (5, 5), input_shape=(100, 100,␣
 ↪3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Dropout(0.05))

model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Dropout(0.05))

model.add(Conv2D(64, (5, 5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
```

(continues on next page)

(continued from previous page)

```
model.add(Dropout(0.05))

model.add(Conv2D(128, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Dropout(0.05))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.05))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.05))

model.add(Dense(num_classes, activation="softmax"))
```

```
model.compile(loss=keras.losses.categorical_
↪crossentropy, optimizer=optimizers.Adam(), metrics=[
↪'accuracy'])
model.fit(train_data, batch_size = batch_size,␣
↪epochs=2)
```

```
Epoch 1/2
6770/6770 [==============================] - 160s␣
↪24ms/step - loss: 1.2582 - accuracy: 0.6622
Epoch 2/2
6770/6770 [==============================] - 129s␣
↪19ms/step - loss: 0.5038 - accuracy: 0.8606
```

```
<tensorflow.python.keras.callbacks.History at␣
↪0x154323500a0>
```

```
score = model.evaluate(train_data)
print(score)
score = model.evaluate(test_data)
print(score)
```

```
6770/6770 [==============================] - 105s␣
↪15ms/step - loss: 0.2151 - accuracy: 0.9366
[0.21505890786647797, 0.9366099238395691]
```

(continues on next page)

(continued from previous page)

```
2269/2269 [==============================] - 34s 15ms/
↪step - loss: 0.8411 - accuracy: 0.8114
[0.8410834670066833, 0.8113980889320374]
```

## 6.5 CNN Architectures

There are various network architectures being used for image classification tasks. VGG16, Inception Net (GoogLeNet), and ResNet are some of the more notable ones as shown in Fig. 6.7.

### *6.5.1 VGG16*

The VGG16 architecture garnered a lot of attention in 2014. It makes the improvement over its predecessor, AlexNet, through replacing large kernel-sized filters (11 and 5 in the first and second convolution layers, respectively) with multiple $3 \times 3$ kernel-sized filters stacked together.

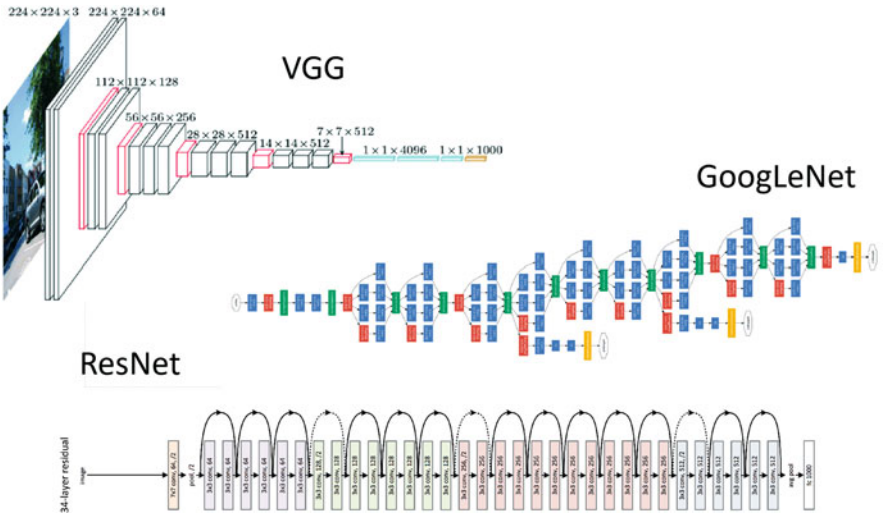



**Fig. 6.7** VGG, Inception V3, and ResNet architectures [3, 7, 8]

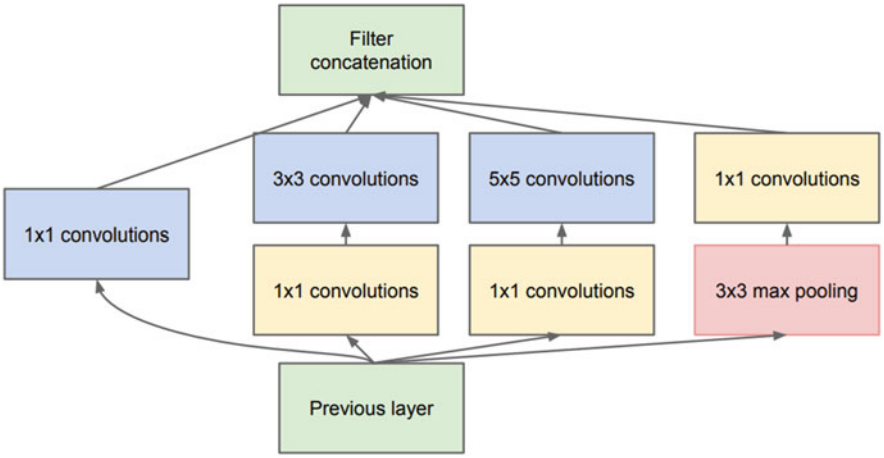



**Fig. 6.8** Illustration of inception module [8]

### *6.5.2 InceptionNet*

Before the dense layers (which are placed at the end of the network), each time we add a new layer we face two main decisions:

(1) Deciding whether we want to go with a pooling or convolution operation
(2) Deciding the size and number of filters to be passed through the output of the previous layer

Google researchers developed the inception module shown in Fig. 6.8 that allows us to apply different options all together in one single layer.

The main idea of the inception module is that of running multiple operations (pooling, convolution) with multiple filter sizes ($3 \times 3$, $5 \times 5 \ldots$) in parallel so that we do not have to face any trade-off.

### *6.5.3 ResNet*

Researchers thought that increasing more layers would improve the accuracy of the models. But there are two problems associated with it:

(1) Vanishing gradient problem—Somewhat solved with regularization like batch normalization, etc. Gradients become increasingly smaller as the network becomes deeper, making it harder to train deep networks.
(2) The authors observed that adding more layers did not improve the accuracy. Also, it is not overfitting also as the training error is also increasing.

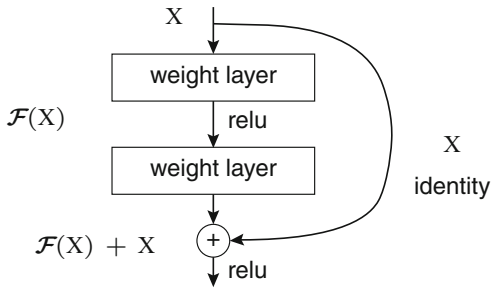



**Fig. 6.9** Illustration of residual connections [3]

Thus, they developed the residual connection as shown in Fig. 6.9. The basic intuition of the residual connections is that, at each convolution layer, the network learns some features about the data F(x) and passes the remaining errors further into the network. So we can say the output error of the convolution layer is H(x) = F(x) -x.

This solution also helped to alleviate the vanishing gradient problem as gradients can flow through the residual connections.

## 6.6 Finetuning

Neural networks are usually initialized with random weights. These weights will converge to some values after training for a series of epochs, to allow us to properly classify our input images. However, instead of a random initialization, we can initialize those weights to values that are already good to classify a different dataset.

Transfer learning is the process of training a network that already performs well on one task, to perform a different task. Finetuning is an example of transfer learning, where we use another network trained on a much larger dataset to initialize and simply train it for classification. In finetuning, we can keep the weights of earlier layers as it has been observed that the early layers contain more generic features, edges, and color blobs and are more common to many visual tasks. Thus we can just finetune the later layers that are more specific to the details of the class.

Through transfer learning, we would not require a dataset as big compared to having to train a network from scratch. We can reduce the required number of images from hundreds of thousands or even millions of images down to just a few thousands. Training time is also sped up during the retraining process as it is much easier due to the initialization.

In the exercise below, we will finetune a ResNet50, pretrained on imagenet (more than 14 million images, consisting of 1000 classes) for the same fruit classification task. In order to speed up the training process, we will freeze ResNet and simply train the last linear layer.

```
from tensorflow.keras.applications.resnet import␣
↪ResNet50
resnet_model = ResNet50(include_top=False, weights=
↪'imagenet', input_shape=(100,100,3))
resnet_model.trainable = False

from tensorflow.keras.layers import Conv2D,␣
↪MaxPooling2D, Flatten, Dense, Dropout, InputLayer,␣
↪GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers
model = Sequential()
model.add(resnet_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_
↪crossentropy, optimizer=optimizers.Adam(), metrics=[
↪'accuracy'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 4, 4, 2048)        23587712
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 131)               268419
=================================================================
Total params: 23,856,131
Trainable params: 268,419
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
model.fit(train_data, epochs=1)
```

```
6770/6770 [==============================] - 388s 57ms/step - loss: 0.1312 -␣
↪accuracy: 0.9728
```

```
<tensorflow.python.keras.callbacks.History at 0x15420d63490>
```

```
score = model.evaluate(train_data)
print(score)
score = model.evaluate(test_data)
print(score)
```

```
6770/6770 [==============================] - 387s 57ms/step - loss: 0.0214 -␣
↪accuracy: 0.9927
[0.021364932879805565, 0.9926578998565674]
2269/2269 [==============================] - 132s 58ms/step - loss: 0.3093 -␣
↪accuracy: 0.9347
[0.3093399107456207, 0.9346791505813599]
```

## 6.7 Other Tasks That Use CNNs

CNNs are used in many other tasks apart from image classification.

### *6.7.1 Object Detection*

Classification tasks only tell us what is in the image and not where the object is. Object detection is the task of localizing objects within an image. CNNs, such as ResNets, are usually used as the feature extractor for object detection networks. An example of an object detection task is shown in Fig. 6.10.

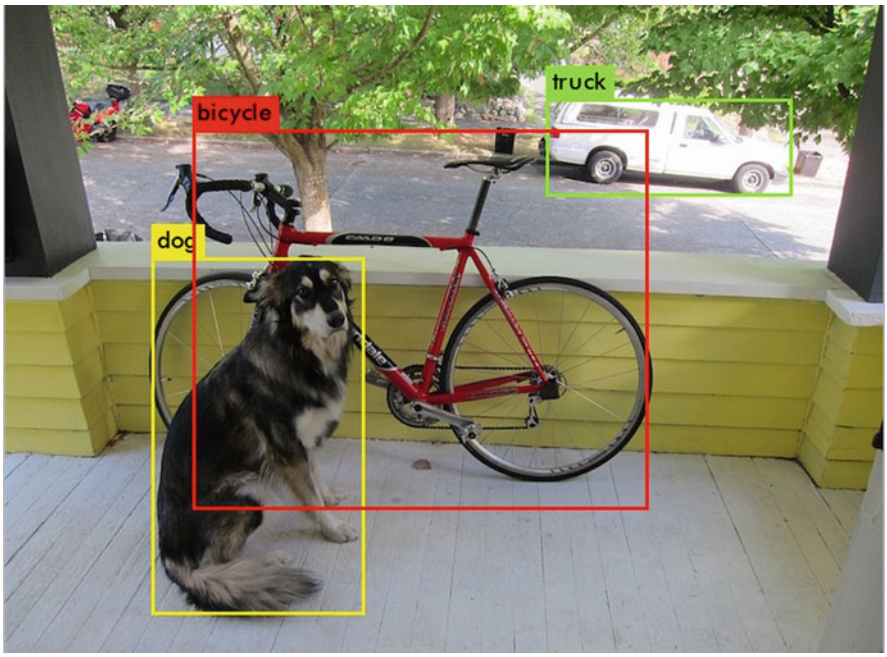



**Fig. 6.10** Example of object detection task

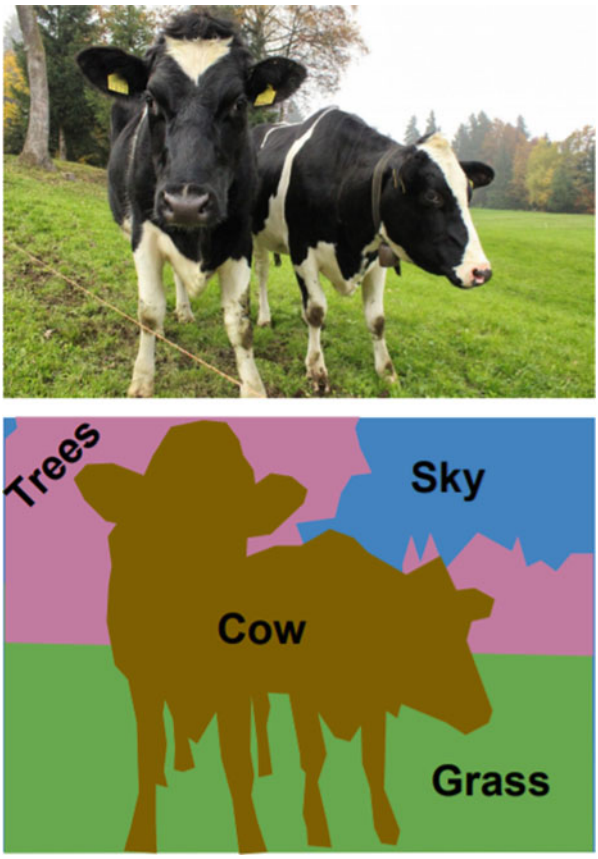



**Fig. 6.11** Example of semantic segmentation task [9]

### *6.7.2 Semantic Segmentation*

Using fully convolutional nets, we can generate output maps that tell us which pixel belongs to which classes. This task, as shown in Fig. 6.11, is called semantic segmentation.

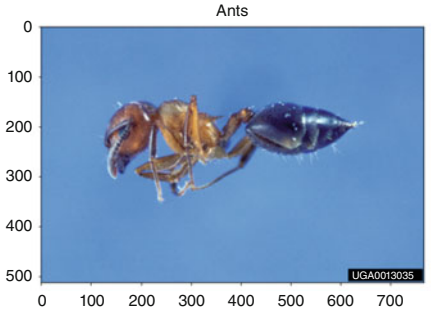
## Exercises

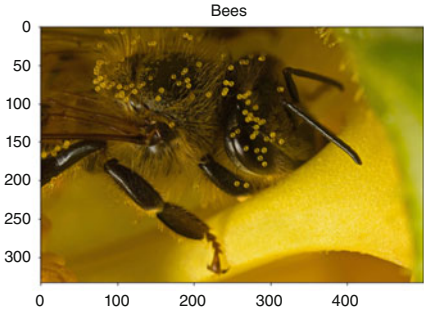
We will utilize this dataset [2]:
https://www.kaggle.com/datasets/gauravduttakiit/ants-bees
This dataset consists of Hymenoptera species, namely ants and bees.

In this exercise, try to develop a convolutional neural network to classify between the two species.
Here you can put what you have learned into practice. Try to:

- Prepare and load the data using ImageDataGenerator in keras.
- Build your own convolutional neural network and train it from scratch.
- Finetune a pretrained CNN for this task.
- Improve the performance of the networks by trying out different parameters such as learning rate.

```
import matplotlib.pyplot as plt
import os
import cv2

ants_folder = "hymenoptera_data/train/ants"
ants_sample = os.path.join(ants_folder, os.
 ↪listdir(ants_folder)[0])
ants_image = cv2.cvtColor(cv2.imread(ants_sample),␣
 ↪cv2.COLOR_BGR2RGB)
fig, axs = plt.subplots(1, 2, figsize = (20,10))
axs[0].imshow(ants_image)
axs[0].set_title("Ants")

bees_folder = "hymenoptera_data/train/bees"
bees_sample = os.path.join(bees_folder, os.
 ↪listdir(bees_folder)[0])
bees_image = cv2.cvtColor(cv2.imread(bees_sample),␣
 ↪cv2.COLOR_BGR2RGB)
axs[1].imshow(bees_image)
axs[1].set_title("Bees")
plt.show()
```



## References


1. Bansal S (2020) Convolutional neural networks explained. https://medium.datadriveninvestor.com/convolutional-neural-networks-explained-7fafea4de9c9
2. Dutta G (2021) Ants and bees. https://www.kaggle.com//datasets/gauravduttakiit/ants-bees
3. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. CoRR abs/1512.03385. http://arxiv.org/abs/1512.03385. 1512.03385
4. Li M (2017) Cs 898: Deep learning and its applications. https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture5CNN.ppt
5. Moltean (2021) Fruits 360. In: Kaggle. https://www.kaggle.com/moltean/fruits. Accessed 27 Apr 2023
6. Potrimba P (2023) What is a convolutional neural network? https://blog.roboflow.com/what-is-a-convolutional-neural-network/
7. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings
8. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2014) Going deeper with convolutions. CoRR abs/1409.4842. http://arxiv.org/abs/1409.4842. 1409.4842
9. University S (2017) Cs231n: Deep learning for computer vision. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf
10. University S (2020) CS231n convolutional neural networks for visual recognition. https://cs231n.github.io/convolutional-networks/

# Chapter 7
# Text Mining


**Learning outcomes:**

- Represent text data in structured and easy-to-consume formats for machine learning and text mining.
- Perform sentence classification tasks on text data.
- Identify important keywords for sentence classification.

Text mining combines both machine learning and natural language processing (NLP) to draw meaning from unstructured text documents. Text mining the driving force behind how a business analyst turns 50,000 hotel guest reviews into specific recommendations, how a workforce analyst improves productivity and reduces employee turnover, and how companies are automating processes using chatbots.

A very popular and current strategy in this field is vectorized term frequency and inverse document frequency (TF-IDF) representation. In fact, Google search engine also uses this technique when a word is searched. It is based on unsupervised learning technique. TF-IDF converts your document text into a bag of words and then assigns a weighted term to each word. In this chapter, we will discuss how to use text mining techniques to get meaningful results for text classification.

## 7.1 Preparing the Data

```
import pandas as pd

#this assumes one json item per line in json file
df=pd.read_json("TFIDF_news.json", lines=True)
```



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_7

```
df.dtypes
```

```
short_description             object
headline                      object
date                  datetime64[ns]
link                          object
authors                       object
category                      object
dtype: object
```

```
#number of rows (datapoints)
len(df)
```

```
124989
```

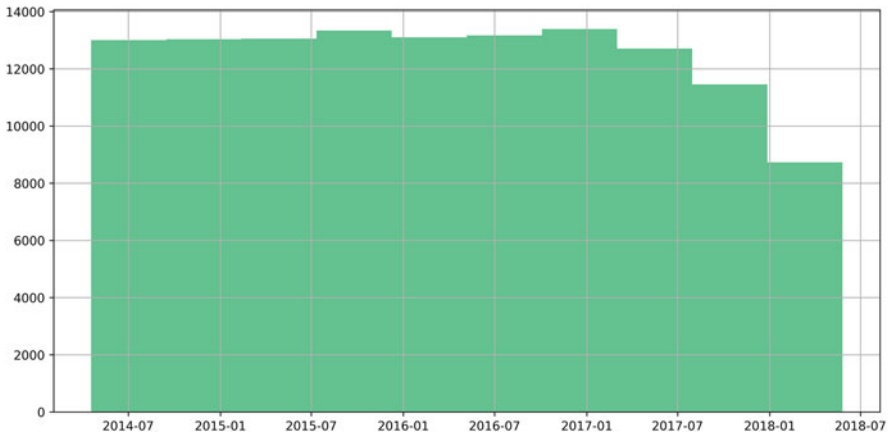
```
# Take sample of 3 to view the data
df.sample(3)
```

```
                                    short_description  \
100659  The hardest battles are not fault in the stree...
74559   Mizzou seems to have catalyzed years of tensio...
48985                     But also hilariously difficult.

                                             headline       date  \
100659     American Sniper Dials in on the Reality of War 2015-01-23
74559   Campus Racism Protests Didn't Come Out Of Nowh... 2015-11-16
48985   These People Took On Puerto Rican Slang And It... 2016-09-02

                                                 link  \
100659  https://www.huffingtonpost.com/entry/american-...
74559   https://www.huffingtonpost.com/entry/campus-ra...
48985   https://www.huffingtonpost.com/entry/these-peo...

                                            authors       category
100659  Zachary Bell, ContributorUnited States Marine ...  ENTERTAINMENT
74559    Tyler Kingkade, Lilly Workneh, and Ryan Grenoble        COLLEGE
48985                                  Carolina Moreno  LATINO VOICES
```

Looking at the range of dates, articles are between July 2014 and July 2018.

```
df.date.hist(figsize=(12,6),color='#86bf91',)
```

```
<matplotlib.axes._subplots.AxesSubplot at␣
↪0x1a695a80508>
```




Before we begin, we need to analyze the distribution of our labels. Looking at the data, there are a total of 31 categories.

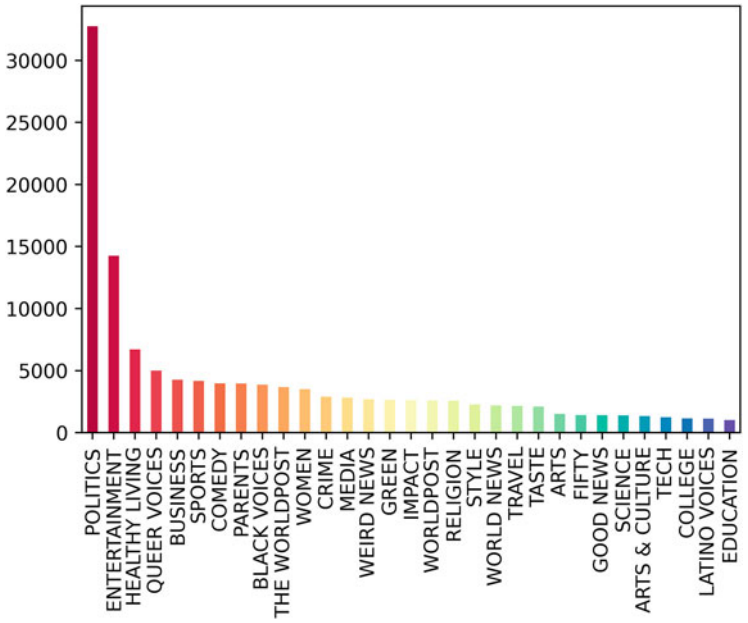
```
len(set(df['category'].values))
```

```
31
```

Most of the articles are related to politics. Education-related articles have the lowest volume.

```
import matplotlib
import numpy as np
cmap = matplotlib.cm.get_cmap('Spectral')
rgba = [cmap(i) for i in np.linspace(0,1,len(set(df[
↪'category'].values)))]
df['category'].value_counts().plot(kind='bar',color␣
↪=rgba)
```

```
<matplotlib.axes._subplots.AxesSubplot at␣
↪0x1a6942753c8>
```



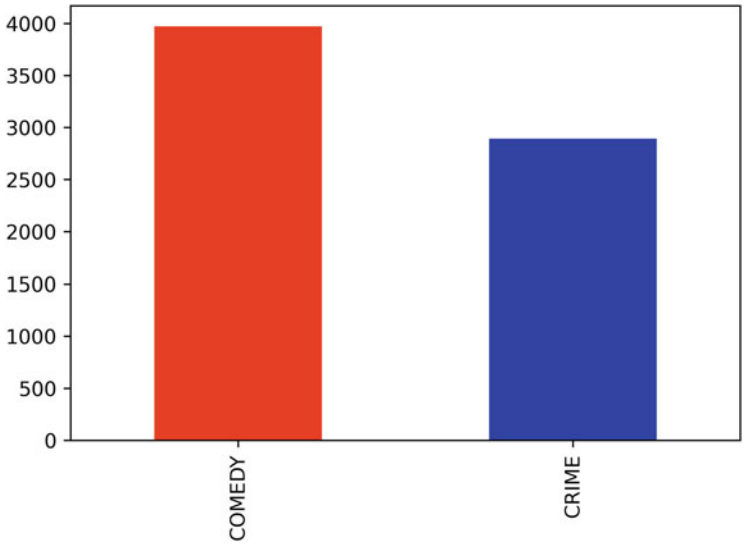

## 7.2 Texts for Classification

In our example, we will only use the headline to predict category. Also, we will only be using 2 categories for simplicity. We will use the `CRIME` and `COMEDY` categories from our dataset.

```
df_orig=df.copy()
df = df_orig[df_orig['category'].isin(['CRIME','COMEDY
↪'])]
print(df.shape)
df.head()
df = df.loc[:, ['headline','category']]
df['category'].value_counts().plot(kind='bar',color =[
↪'r','b'])
```

```
(6864, 6)
```

```
<matplotlib.axes._subplots.AxesSubplot at␣
↪0x1a695c76388>
```




## 7.3 Vectorize

Text vectorization is the process of converting text into numerical representation. Some of the more commonly used techniques for text vectorization are:

- Binary term frequency
- Bag of words (BoW) term frequency
- (L1) Normalized term frequency
- (L2) Normalized TF-IDF
- Word2Vec

**Binary Term Frequency**
Binary term frequency captures the presence (1) or absence (0) of term in document. For this part, under TfidfVectorizer, we set binary parameter equal to true so that it can show just presence or absence.

**Bag of Words (BoW) Term Frequency**
Bag of words (BoW) term frequency captures frequency of term in document. Under TfidfVectorizer, we set binary parameter equal to false so that it can show the actual frequency of the term and norm parameter equal to none.

The following code is an example of bag of words term frequency:

```
from sklearn.feature_extraction.text import␣
↪CountVectorizer

sample_doc = ["Hello I am a boy", "Hello I am a␣
↪student", "My name is Jill"]
cv=CountVectorizer(max_df=0.85)
word_count_vector=cv.fit_transform(sample_doc)
word_count_vector_arr = word_count_vector.toarray()
pd.DataFrame(word_count_vector_arr,columns=sorted(cv.
↪vocabulary_, key=cv.vocabulary_.get))
```

| | am | boy | hello | is | jill | my | name | student |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

An important note is the vocabulary is placed in a dictionary and python dictionaries are unsorted. Notice that the header in the following code is different from the first example.

```
## Wrong example
pd.DataFrame(word_count_vector_arr,columns=cv.
↪vocabulary_)
```

| | hello | am | boy | student | my | name | is | jill |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

This is because of dictionary in Python. See below:

```
cv.vocabulary_
```

```
{'hello': 2,
 'am': 0,
 'boy': 1,
 'student': 7,
 'my': 5,
 'name': 6,
 'is': 3,
 'jill': 4}
```

Let us move on to our code example. Now, let us look at 10 words from our vocabulary. We have also removed words that appear in 95% of the documents. In text analytics, such words (stop words) are not meaningful. An intuitive approach to understanding removal of stop words is that in a sentence, many words are present because of grammatical rules and do not add extra content or meaning. Ignoring such words would allow us to distill the key essence of a document and sentence. Sweet, after removing stop words by having `maxdf=0.95`, our keywords are mostly crime and comedy related.

```
from sklearn.feature_extraction.text import␣
↪CountVectorizer
docs=df['headline'].tolist()
# create a vocabulary of words,
# ignore words that appear in 85% of documents,
# eliminate stop words
cv=CountVectorizer(max_df=0.95)
word_count_vector=cv.fit_transform(docs)
list(cv.vocabulary_.keys())[:10]
```

```
['there',
 'were',
 'mass',
 'shootings',
 'in',
 'texas',
 'last',
 'week',
 'but',
 'only']
```

We can also use machine learning models learned previously to classify our headlines! See code below:

```
from sklearn.feature_extraction.text import␣
↪TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

df['category_is_crime'] = df['category']=='CRIME'
X_train, X_test, y_train, y_test = train_test_
↪split(word_count_vector, df['category_is_crime'],␣
↪test_size=0.2, random_state=42)
```

Wow, we achieve 95.19% in classifying headlines. This is a remarkable feat for our machine!

```
model1 = LogisticRegression()
model1.fit(X_train, y_train)

y_pred = model1.predict(X_test)
cm=confusion_matrix(y_test, y_pred)
print(cm)
acc=(cm[0,0]+cm[1,1])/sum(sum(cm))
print('Accuracy of a simple linear model with TFIDF␣
↪is .... {:.2f}%'.format(acc*100))
```

```
[[766  26]
 [ 40 541]]
Accuracy of a simple linear model with TF-IDF is ....␣
↪95.19%
```

## 7.4 TF-IDF

```
tfidf_transformer=TfidfTransformer(smooth_idf=True,
↪use_idf=True)
tfidf_x_train = tfidf_transformer.fit_transform(X_
↪train)
model1 = LogisticRegression()
model1.fit(tfidf_x_train, y_train)
tfidf_x_test = tfidf_transformer.transform(X_test)
y_pred = model1.predict(tfidf_x_test)
cm=confusion_matrix(y_test, y_pred)
print(cm)
acc=(cm[0,0]+cm[1,1])/sum(sum(cm))
print('Accuracy of a simple linear model with TFIDF␣
↪is .... {:.2f}%'.format(acc*100))
```

```
[[777  15]
 [ 57 524]]
Accuracy of a simple linear model with TF-IDF is ....␣
↪94.76%
```

Apart from text classification, we can use TF-IDF to discover “important” keywords. Here are a few examples that show the importance of each individual

word. Such technique is simple and easy to use. But on a cautionary note, using TF-IDF is heavily dependent on the input data, and the importance of the text is closely related to the frequency in the document and across the entire data.

```
## Important keywords extraction using tfidf
print(df.iloc[1].headline)
vector = cv.transform([df.iloc[1].headline])
tfidf_vector = tfidf_transformer.transform(vector)
coo_matrix = tfidf_vector.tocoo()
tuples = zip(coo_matrix.col, coo_matrix.data)
sorted_tuple = sorted(tuples, key=lambda x: (x[1],␣
↪x[0]), reverse=True)
[(cv.get_feature_names()[i[0]],i[1]) for i in sorted_
↪tuple]
```

```
Rachel Dolezal Faces Felony Charges For Welfare Fraud
```

```
[('welfare', 0.413332601468908),
 ('felony', 0.413332601468908),
 ('dolezal', 0.413332601468908),
 ('rachel', 0.3885287853920158),
 ('fraud', 0.3599880238280249),
 ('faces', 0.3103803916742406),
 ('charges', 0.2954500640160872),
 ('for', 0.15262948420298186)]
```

```
## Important keywords extraction using tfidf
print(df.iloc[5].headline)
vector = cv.transform([df.iloc[5].headline])
tfidf_vector = tfidf_transformer.transform(vector)
coo_matrix = tfidf_vector.tocoo()
tuples = zip(coo_matrix.col, coo_matrix.data)
sorted_tuple = sorted(tuples, key=lambda x: (x[1],␣
↪x[0]), reverse=True)
[(cv.get_feature_names()[i[0]],i[1]) for i in sorted_
↪tuple]
```

```
Man Faces Charges After Pulling Knife, Stun Gun On␣
↪Muslim Students At McDonald's
```

```
[('stun', 0.37604716794652987),
 ('pulling', 0.3658447343442784),
```

(continues on next page)

(continued from previous page)

```
 ('knife', 0.32581708572483403),
 ('mcdonald', 0.32215742177499496),
 ('students', 0.30480662832662847),
 ('faces', 0.2922589939460096),
 ('muslim', 0.28707744879148683),
 ('charges', 0.27820036570239326),
 ('gun', 0.24718607863715278),
 ('at', 0.17925932409191916),
 ('after', 0.17428789091260877),
 ('man', 0.17199120825269787),
 ('on', 0.15323370190782204)]
```

```
comedy_1 = df[~df['category_is_crime']].iloc[0].
↪headline
print(comedy_1)
```

```
Trump's New 'MAGA'-Themed Swimwear Sinks On Twitter
```

```
## Important keywords extraction using tfidf
vector = cv.transform([comedy_1])
tfidf_vector = tfidf_transformer.transform(vector)
coo_matrix = tfidf_vector.tocoo()
tuples = zip(coo_matrix.col, coo_matrix.data)
sorted_tuple = sorted(tuples, key=lambda x: (x[1],␣
↪x[0]), reverse=True)
[(cv.get_feature_names()[i[0]],i[1]) for i in sorted_
↪tuple]
```

```
[('swimwear', 0.4735563110982704),
 ('sinks', 0.4735563110982704),
 ('maga', 0.4735563110982704),
 ('themed', 0.37841071080711314),
 ('twitter', 0.2770106227768904),
 ('new', 0.22822300865931006),
 ('on', 0.17796879475963143),
 ('trump', 0.15344404805174222)]
```

## 7.5 Web Scraping

The BeautifulSoup4 package is used to send requests to various websites and return their html page. From there, you can search and extract relevant text information to perform analysis on.

```
import requests
from bs4 import BeautifulSoup

page = requests.get("http://www.facebook.com")
soup = BeautifulSoup(page.content, "html.parser")

print(soup)
```

## 7.6 Tokenization

Tokenization is used to break down larger chunks of text into smaller blocks called tokens. This allows us to assign a meaning to these smaller building blocks that can be reused in various documents.

```
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer()
s0 = "This is a cooool #dummysmiley: :-) :-P <3 and␣
↪some arrows < > -> <--"
print(tknzr.tokenize(s0))
```

```
['This', 'is', 'a', 'cooool', '#dummysmiley', ':', ':-
↪)', ':-P', '<3', 'and', 'some', 'arrows', '<', '>',
↪'->', '<--']
```

## 7.7 Part of Speech Tagging

Part of speech tagging is used to label each word with a corresponding part of speech tag that describes how the word was used in a sentence. This includes nouns, verbs, adjectives, adverbs, and more. We can use the averaged perceptron tagger to tag our tokens with corresponding part of speech tags.

```
import nltk
nltk.download('averaged_perceptron_tagger')
```

(continues on next page)

(continued from previous page)

```
nltk.download("tagsets")
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer()
s0 = "This is a cooool #dummysmiley: :-) :-P <3 and␣
↪some arrows < > -> <--"
tokens=tknzr.tokenize(s0)
tagged = nltk.pos_tag(tokens)
print(tagged)
```

```
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('cooool
↪', 'JJ'), ('#dummysmiley', 'NN'), (':', ':'), (':-)
↪', 'JJ'), (':-P', 'JJ'), ('<3', 'NN'), ('and', 'CC
↪'), ('some', 'DT'), ('arrows', 'NNS'), ('<', 'VBP'),
↪ ('>', 'JJ'), ('->', 'CD'), ('<--', 'JJ')]
```

## 7.8 Stemming and Lemmatization

Stemming removes common suffixes in an attempt to convert words back into their root words. For example, the four words, marketing, markets, marketed, and marketer, all have market as the root word. Thus stemming is used to convert all of them into market.

```
from nltk.stem import PorterStemmer

ps = PorterStemmer()

sample_words = ["marketing", "markets", "marketed",
↪"marketer"]

print(sample_words)

for each in sample_words:
    print("{:s} -> {:s}".format(each, ps.stem(each)))
```

```
['marketing', 'markets', 'marketed', 'marketer']
marketing -> market
markets -> market
marketed -> market
marketer -> market
```

However, this may not always work out well. For example, words such as busy and business will end up mapping to busi but have two completely different meanings. On the other hand, lemmatization handles this properly by converting words into their meaningful base form otherwise known as the lemma. It requires the use of a vocabulary and morphological analysis of the words used.

```
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')
wnl = WordNetLemmatizer()

print(wnl.lemmatize("beaten"))
print(wnl.lemmatize("beaten", "v"))
print(wnl.lemmatize("women", "n"))
print(wnl.lemmatize("happiest", "a"))
```

```
beaten
beat
woman
happy
```

## Exercises

We will utilize this dataset [1]:
https://www.kaggle.com/datasets/team-ai/spam-text-message-classification

This dataset consists of SMS text messages that are labeled as spam or not spam (labeled as spam and ham, respectively).

In this exercise, try to develop a text classification model to classify between the two types of text.

Here you can put what you have learned into practice. Try to:

- Analyze the label distribution.
- Experiment with various text vectorizers covered in this chapter with various classification models.

```
import pandas as pd
df = pd.read_csv('SPAM text message 20170820 - Data.
↪csv')
df["Spam"] = df["Category"] == "spam"
df.drop("Category", axis=1, inplace=True)
df.head()
```

```
Message   Spam
0  Go until jurong point, crazy.. Available only ...  False
1                      Ok lar... Joking wif u oni...  False
2  Free entry in 2 a wkly comp to win FA Cup fina...   True
3  U dun say so early hor... U c already then say...  False
4  Nah I don't think he goes to usf, he lives aro...  False
```

## Reference

1. AI T (2017) Spam text message classification. https://www.kaggle.com/datasets/team-ai/spam-text-message-classification

# Chapter 8
# Chatbot, Speech, and NLP


**Learning Outcomes**

- Explore into speech to text capabilities in Python.
- Represent text data in structured and easy-to-consume formats for chatbots.
- Familiarize with the encoder–decoder architecture.
- Develop a chatbot to answer questions.

In this chapter, we will explore the speech to text capabilities with Python, and then we will assemble a seq2seq long short-term memory (LSTM) model using Keras Functional API to create an example chatbot that would answer questions asked to it. You can try integrating both programs together. However, do note that the code we have provided does not integrate both components.

Chatbots have become applications themselves. You can choose the field or stream and gather data regarding various questions. We can build a chatbot for an e-commerce website or a school website where parents could get information about the school.

Messaging platforms such as Allo have implemented chatbot services to engage users. The famous https://assistant.google.com/Google Assistant, https://www.apple.com/in/siri/Siri, https://www.microsoft.com/en-in/windows/cortanaCortana, and https://www.alexa.com/Alexa were built using similar models.

So, let us start building our chatbot.

## 8.1 Speech to Text

```
#pip install SpeechRecognition
#pip install pipwin
```

(continues on next page)



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_8

(continued from previous page)

```
#pipwin install pyaudio
import speech_recognition as sr
import sys
r = sr.Recognizer()

print("please say something in 4 seconds... and wait␣
↪for 4 seconds for the answer.....")
print("Accessing Microphone..")

try:
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source, duration=2)
    # use the default microphone as the audio source,␣
↪duration higher means environment noisier
        print("Waiting for you to speak...")
        audio = r.listen(source)                    #␣
↪listen for the first phrase and extract it into␣
↪audio data

except (ModuleNotFoundError,AttributeError):
    print('Please check installation')
    sys.exit(0)

try:
    print("You said " + r.recognize_google(audio))
↪# recognize speech using Google Speech Recognition
except LookupError:                                 #␣
↪speech is unintelligible
    print("Could not understand audio")

except:
    print("Please retry...")
```

```
please say something in 4 seconds... and wait for 4 seconds for the answer.....
Accessing Microphone..
Waiting for you to speak...
result2:
{   'alternative': [   {'confidence': 0.97219545, 'transcript': 'hello hello'},
                       {'transcript': 'hello hello hello'}],
    'final': True}
You said hello hello
```

Here is how we can run the same speech to text function using pre-recorded wave files as well:

```
from IPython.display import Audio
import speech_recognition as sr
import sys
r = sr.Recognizer()

with sr.WavFile("Welcome.wav") as source:
    audio = r.record(source)

try:
    text = r.recognize_google(audio)
    print("You said: " + text)
except LookupError:
    print("Could not understand audio")
```

```
You said: thank you for choosing the Olympus␣
↪dictation management system the Olympus dictation␣
↪management system gives you the power to manage␣
↪your dictations transcriptions and documents␣
↪seamlessly and to improve the productivity of your␣
↪daily work for example you can automatically send␣
↪the dictation files or transcribe documents to your␣
↪assistant or the author via email or FTP if you're␣
↪using the speech recognition software the speech␣
↪recognition engine works in the background to␣
↪support your document creation we hope you enjoy␣
↪the simple flexible reliable and Secure Solutions␣
↪from Olympus
```

The following code requires `SpeechRecognition`, `pipwin`, and `pyaudio`. Please install first before carrying out the code. The code adjusts according to the ambient noise that helps to capture what we have said. If there is an error, try adjusting the `duration` parameter in `adjust_for_ambient_noise`. Now we have managed to capture what is said in `r.recognize_google(audio)`. We will be able to use this can pass it to our chatbot.

```
import numpy as np
import tensorflow as tf
import pickle
from tensorflow.keras import layers , activations ,␣
↪models , preprocessing
```

## 8.2 Preparing the Data for Chatbot

### *8.2.1 Download the Data*

The dataset we will be using comes from this link [1]:
https://github.com/shubham0204/Dataset_Archives/blob/master/chatbot_nlp.zip?raw=true

This dataset contains pairs of questions and answers based on a number of subjects such as food, history, AI, etc.

To start, we will need to download the dataset and unzip the file. There should be a chatbot_nlp folder that contains the data we will be using.

### *8.2.2 Reading the Data from the Files*

We will import https://www.tensorflow.orgTensorFlow and our beloved https://www.tensorflow.org/guide/kerasKeras. Also, we import other modules that help in defining model layers.

We parse each of the `.yaml` files:

- Concatenate two or more sentences if the answer has two or more of them.
- Remove unwanted data types that are produced while parsing the data.
- Append `<START>` and `<END>` to all the `answers`.
- Create a `Tokenizer` and load the whole vocabulary ( `questions + answers` ) into it.

```
from tensorflow.keras import preprocessing , utils
import os
import yaml


dir_path = 'chatbot_nlp/data'
files_list = os.listdir(dir_path + os.sep)

questions = list()
answers = list()

for filepath in files_list:
    stream = open( dir_path + os.sep + filepath , 'rb
↪')
    docs = yaml.safe_load(stream)
    conversations = docs['conversations']
```

(continues on next page)

(continued from previous page)

```
    for con in conversations:
        if len( con ) > 2 :
            questions.append(con[0])
            replies = con[ 1 : ]
            ans = ''
            for rep in replies:
                ans += ' ' + rep
            answers.append( ans )
        elif len( con )> 1:
            questions.append(con[0])
            answers.append(con[1])

answers_with_tags = list()
for i in range( len( answers ) ):
    if type( answers[i] ) == str:
        answers_with_tags.append( answers[i] )
    else:
        questions.pop( i )

answers = list()
for i in range( len( answers_with_tags ) ) :
    answers.append( '<START> ' + answers_with_tags[i]␣
↪+ ' <END>' )

tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts( questions + answers )
VOCAB_SIZE = len( tokenizer.word_index )+1
print( 'VOCAB SIZE : {}'.format( VOCAB_SIZE ))
```

```
VOCAB SIZE : 1894
```

### 8.2.3 *Preparing Data for Seq2Seq Model*

Our model requires three arrays namely `encoder_input_data`, `decoder_input_data`, and `decoder_output_data`.

For `encoder_input_data`:

- Tokenize the `questions`. Pad them to their maximum length.

For `decoder_input_data`:

- Tokenize the `answers`. Pad them to their maximum length.

For `decoder_output_data`:

- Tokenize the `answers`. Remove the first element from all the `tokenized_answers`. This is the `<START>` element which we added earlier.

```
from gensim.models import Word2Vec
import re

vocab = []
for word in tokenizer.word_index:
    vocab.append( word )

def tokenize( sentences ):
    tokens_list = []
    vocabulary = []
    for sentence in sentences:
        sentence = sentence.lower()
        sentence = re.sub( '[^a-zA-Z]', ' ', sentence␣
↪)
        tokens = sentence.split()
        vocabulary += tokens
        tokens_list.append( tokens )
    return tokens_list , vocabulary

# encoder_input_data
tokenized_questions = tokenizer.texts_to_sequences(␣
↪questions )
maxlen_questions = max( [ len(x) for x in tokenized_
↪questions ] )
padded_questions = preprocessing.sequence.pad_
↪sequences( tokenized_questions , maxlen=maxlen_
↪questions , padding='post' )
encoder_input_data = np.array( padded_questions )
print( encoder_input_data.shape , maxlen_questions )


# decoder_input_data
tokenized_answers = tokenizer.texts_to_sequences(␣
↪answers )
maxlen_answers = max( [ len(x) for x in tokenized_
↪answers ] )
padded_answers = preprocessing.sequence.pad_
↪sequences( tokenized_answers , maxlen=maxlen_
↪answers , padding='post' )
```

(continues on next page)

(continued from previous page)

```
decoder_input_data = np.array( padded_answers )
print( decoder_input_data.shape , maxlen_answers )

# decoder_output_data
tokenized_answers = tokenizer.texts_to_sequences( answers )
for i in range(len(tokenized_answers)) :
    tokenized_answers[i] = tokenized_answers[i][1:]
padded_answers = preprocessing.sequence.pad_sequences( tokenized_answers , maxlen=maxlen_answers , padding='post' )
onehot_answers = utils.to_categorical( padded_answers, VOCAB_SIZE )
decoder_output_data = np.array( onehot_answers )
print( decoder_output_data.shape )
```

```
(564, 22) 22
(564, 74) 74
(564, 74, 1894)
```

```
tokenized_questions[0],tokenized_questions[1]
```

```
([10, 7, 269], [10, 7, 269])
```

```
padded_questions[0].shape
```

```
(22,)
```

## 8.3 Defining the Encoder–Decoder Model

The model will have embedding, LSTM, and dense layers. The basic configuration is as shown in Fig. 8.1:

- 2 Input Layers: One for `encoder_input_data` and another for `decoder_input_data`.
- Embedding layer: For converting token vectors to fix sized dense vectors. (Note: Do not forget the `mask_zero=True` argument here. )
- LSTM layer: Provide access to long short-term cells.

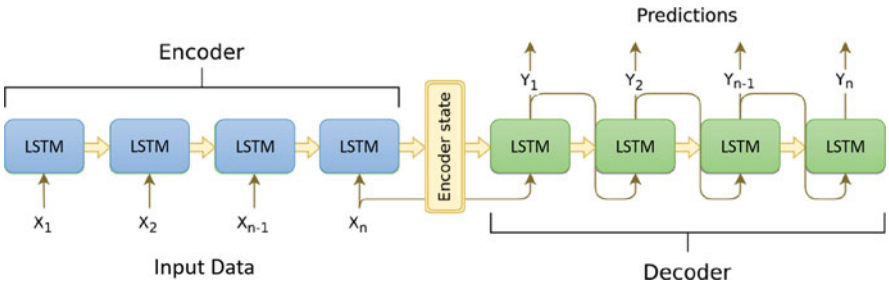



**Fig. 8.1** Illustration of the Seq2Seq architecture

Working:

1. The `encoder_input_data` comes in the embedding layer (`encoder_embedding` ).
2. The output of the embedding layer goes to the LSTM cell, which produces 2 state vectors ***( `h` and `c` that are `encoder_states` ).
3. These states are set in the LSTM cell of the decoder.
4. The decoder_input_data comes in through the embedding layer.
5. The embeddings go into LSTM cell (which had the states) to produce sequences.

```
encoder_inputs = tf.keras.layers.Input(shape=( maxlen_
↪questions , ))
encoder_embedding = tf.keras.layers.Embedding( VOCAB_
↪SIZE, 200 , mask_zero=True ) (encoder_inputs)
encoder_outputs , state_h , state_c = tf.keras.layers.
↪LSTM( 200 , return_state=True )( encoder_embedding )
encoder_states = [ state_h , state_c ]

decoder_inputs = tf.keras.layers.Input(shape=( maxlen_
↪answers ,  ))
decoder_embedding = tf.keras.layers.Embedding( VOCAB_
↪SIZE, 200 , mask_zero=True) (decoder_inputs)
decoder_lstm = tf.keras.layers.LSTM( 200 , return_
↪state=True , return_sequences=True )
decoder_outputs , _ , _ = decoder_lstm ( decoder_
↪embedding , initial_state=encoder_states )
decoder_dense = tf.keras.layers.Dense( VOCAB_SIZE ,␣
↪activation=tf.keras.activations.softmax )
output = decoder_dense ( decoder_outputs )

model = tf.keras.models.Model([encoder_inputs,␣
↪decoder_inputs], output )
```

(continues on next page)

(continued from previous page)

```
model.compile(optimizer=tf.keras.optimizers.RMSprop(),
 ↪ loss='categorical_crossentropy')

model.summary()
```

```
Model: "model"
__________________________________________________________________________________
↪________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)            [(None, 22)]         0
__________________________________________________________________________________
↪________________
input_2 (InputLayer)            [(None, 74)]         0
__________________________________________________________________________________
↪________________
embedding (Embedding)           (None, 22, 200)      378800      input_1[0][0]
__________________________________________________________________________________
↪________________
embedding_1 (Embedding)         (None, 74, 200)      378800      input_2[0][0]
__________________________________________________________________________________
↪________________
lstm (LSTM)                     [(None, 200), (None, 320800      embedding[0][0]
__________________________________________________________________________________
↪________________
lstm_1 (LSTM)                   [(None, 74, 200), (N 320800      embedding_1[0][0]
                                                                 lstm[0][1]
                                                                 lstm[0][2]
__________________________________________________________________________________
↪________________
dense (Dense)                   (None, 74, 1894)     380694      lstm_1[0][0]
==================================================================================
Total params: 1,779,894
Trainable params: 1,779,894
Non-trainable params: 0
__________________________________________________________________________________
↪________________
```

## 8.4 Training the Model

We train the model for a number of epochs with `RMSprop` optimizer and `categorical_crossentropy` loss function.

```
model.fit([encoder_input_data , decoder_input_data],␣
 ↪decoder_output_data, batch_size=50, epochs=150,␣
 ↪verbose=0 )
model.save( 'model.h5' )
```

```
output = model.predict([encoder_input_data[0,np.
 ↪newaxis], decoder_input_data[0,np.newaxis]])
```

```
output[0][0]
```

```
array([4.5021617e-10, 2.7967335e-06, 2.6644248e-08, ..., 7.4593916e-12,
       6.7842798e-08, 1.0224695e-08], dtype=float32)
```

```
np.argmax(output[0][0])
```

```
132
```

```
import pandas as pd
```

```
tokenizer_dict = { tokenizer.word_index[i]:i for i in␣
↪tokenizer.word_index}
pd.DataFrame.from_dict(tokenizer_dict, orient='index
↪').tail(15)
```

```
                0
1879          par
1880         dont
1881      richard
1882        nixon
1883         1963
1884       soviet
1885        union
1886      sputnik
1887    gyroscope
1888        edwin
1889    andromeda
1890      kingdom
1891      britain
1892       europe
1893  echolocation
```

```
tokenizer_dict[np.argmax(output[0][1])]
```

```
'intelligence'
```

```
tokenizer_dict[np.argmax(output[0][2])]
```

```
'is'
```

```
output = model.predict([encoder_input_data[0,np.
↪newaxis], decoder_input_data[0,np.newaxis]])
sampled_word_indexes = np.argmax(output[0],1)
sentence = ""
maxlen_answers = 74
for sampled_word_index in sampled_word_indexes:
    sampled_word = None
    sampled_word = tokenizer_dict[sampled_word_index]
    if sampled_word == 'end' or len(sentence.split())␣
↪> maxlen_answers:
        break
    sentence += ' {}'.format( sampled_word )
sentence
```

```
' artificial intelligence is the branch of␣
↪engineering and science devoted to constructing␣
↪machines that think'
```

```
def print_train_result(index):
    print(f"Question is : {questions[index]}")
    print(f"Answer is : {answers[index]}")
    output = model.predict([encoder_input_data[index,
↪np.newaxis], decoder_input_data[index,np.newaxis]])
    sampled_word_indexes = np.argmax(output[0],1)
    sentence = ""
    maxlen_answers = 74
    for sampled_word_index in sampled_word_indexes:
        sampled_word = None
        sampled_word = tokenizer_dict[sampled_word_
↪index]
        if sampled_word == 'end' or len(sentence.
↪split()) > maxlen_answers:
            break
        sentence += ' {}'.format( sampled_word )
    print(f"Model prediction: {sentence}")
```

```
print_train_result(4)
```

```
Question is : Are you sentient?
Answer is : <START> Even though I'm a construct I do␣
 ↪have a subjective experience of the universe, as␣
 ↪simplistic as it may be. <END>
Model prediction:  even though i'm a construct i do␣
 ↪have a subjective experience of the universe as␣
 ↪simplistic as it may be
```

```
print_train_result(55)
```

```
Question is : What is a motormouth
Answer is : <START> A ratchet-jaw. <END>
Model prediction:  a ratchet jaw
```

```
print_train_result(32)
```

```
Question is : Bend over
Answer is : <START> That's personal! <END>
Model prediction:  that's personal
```

## 8.5 Defining Inference Models

We create inference models that help in predicting answers.

**Encoder Inference Model** Takes the question as input and outputs LSTM states (`h` and `c`).

**Decoder Inference Model** Takes in 2 inputs, one are the LSTM states (output of encoder model), and second are the answer input sequences (ones not having the `<start>` tag). It will output the answers for the question which we fed to the encoder model and its state values.

```
def make_inference_models():

    encoder_model = tf.keras.models.Model(encoder_
 ↪inputs, encoder_states)

    decoder_state_input_h = tf.keras.layers.
 ↪Input(shape=( 200 ,))
    decoder_state_input_c = tf.keras.layers.
 ↪Input(shape=( 200 ,))
```

(continues on next page)

(continued from previous page)

```
    decoder_states_inputs = [decoder_state_input_h,␣
↪decoder_state_input_c]

    decoder_outputs, state_h, state_c = decoder_lstm(
        decoder_embedding , initial_state=decoder_
↪states_inputs)
    decoder_states = [state_h, state_c]
    decoder_outputs = decoder_dense(decoder_outputs)
    decoder_model = tf.keras.models.Model(
        [decoder_inputs] + decoder_states_inputs,
        [decoder_outputs] + decoder_states)

    return encoder_model , decoder_model
```

## 8.6 Talking with Our Chatbot

First, we define a method `str_to_tokens` that converts `str` questions into integer tokens with padding.

```
def str_to_tokens( sentence : str ):
    words = sentence.lower().split()
    tokens_list = list()
    for word in words:
        tokens_list.append( tokenizer.word_index[ word ] )
    return preprocessing.sequence.pad_sequences( [tokens_list] , maxlen=maxlen_
↪questions , padding='post')
```

1. First, we take a question as input and predict the state values using `enc_model`.
2. We set the state values in the decoder's LSTM.
3. Then, we generate a sequence that contains the `<start>` element.
4. We input this sequence in the `dec_model`.
5. We replace the `<start>` element with the element that was predicted by the `dec_model` and update the state values.
6. We carry out the above steps iteratively till we hit the `<end>` tag or the maximum answer length.

```
enc_model , dec_model = make_inference_models()


states_values = enc_model.predict( str_to_tokens(␣
↪input( 'Enter question : ' ) ) )
```

(continues on next page)

(continued from previous page)

```
empty_target_seq = np.zeros( ( 1 , 1 ) )
empty_target_seq[0, 0] = tokenizer.word_index['start']
stop_condition = False
decoded_translation = ''
while not stop_condition :
    dec_outputs , h , c = dec_model.predict([ empty_
↪target_seq ] + states_values )
    sampled_word_index = np.argmax( dec_outputs[0, -1,
↪ :] )
    sampled_word = None
    for word , index in tokenizer.word_index.items() :
        if sampled_word_index == index :
            sampled_word = word
            if sampled_word == 'end' or len(decoded_
↪translation.split()) > maxlen_answers:
                stop_condition = True
                break
            decoded_translation += ' {}'.format( word␣
↪)

    empty_target_seq = np.zeros( ( 1 , 1 ) )
    empty_target_seq[ 0 , 0 ] = sampled_word_index
    states_values = [ h , c ]

print( decoded_translation )
```

```
Enter question : what is capitalism
```

```
 the economic system in which all or most of the␣
↪means of production and distribution as land␣
↪factories railroads etc are privately owned and␣
↪operated for profit originally under fully␣
↪competitive conditions
```

Now we have a working demonstration of a chatbot that has been trained on our data. This example is a simple demonstration of how chatbots are trained, as it will face difficulties with words not in our dataset and also may not always give great answers. Chatbots such as Siri, Alexa, and Google Assistant are created with a larger vocabulary, significantly more data, and more complex architectures, which help them perform significantly better than the example provided.

## Exercises

We will utilize this dataset [2]:
https://www.kaggle.com/datasets/rtatman/questionanswer-dataset/

This dataset consists of various Question and Answer pairs about many different topics.

In this exercise, try to develop a chatbot that can answer questions provided in the dataset.

Here you can put what you have learned into practice. Try to:

- Perform data preparation by converting the Questions and Answer into tokens.
- Consider cleaning up the Questions and Answers using the techniques learned.
- Experiment with various network layer configurations such as increasing the number of hidden layers.
- Touch up the interface by adding a speech to text module for inputs and text to speech for outputs.

```
import pandas as pd
df = pd.read_csv('S08_question_answer_pairs.txt',␣
↪delimiter="\t")
df.head()
```

```
ArticleTitle                                          Question Answer  \
0  Abraham_Lincoln  Was Abraham Lincoln the sixteenth President of...    yes
1  Abraham_Lincoln  Was Abraham Lincoln the sixteenth President of...   Yes.
2  Abraham_Lincoln  Did Lincoln sign the National Banking Act of 1...    yes
3  Abraham_Lincoln  Did Lincoln sign the National Banking Act of 1...   Yes.
4  Abraham_Lincoln                  Did his mother die of pneumonia?     no

DifficultyFromQuestioner DifficultyFromAnswerer  ArticleFile
0                      easy                   easy  S08_set3_a4
1                      easy                   easy  S08_set3_a4
2                      easy                 medium  S08_set3_a4
3                      easy                   easy  S08_set3_a4
4                      easy                 medium  S08_set3_a4
```

## References

1. Kausr25 (2019) Chatterbot English. In: Kaggle. https://www.kaggle.com/datasets/kausr25/chatterbotenglish. Accessed 27 Apr 2023
2. Tatman R (2019) Question-answer dataset. https://www.kaggle.com/datasets/rtatman/questionanswer-dataset

# Part II
# Applications of Artificial Intelligence in Business Management

# Chapter 9
# AI in Human Resource Management


**Learning Outcomes**

- Understand the roles and responsibilities of human resource departments.
- Be able to list some of the challenges that human resource departments face.
- Identify ways that artificial intelligence can be applied to human resource management.
- Develop artificial intelligence solutions for recommending salaries, recruitment, recommending courses, and predicting attrition.

## 9.1 Introduction to Human Resource Management

The human resource department is vital in managing one of the most precious resources in all businesses and its employees. Human resources is in charge of recruiting, onboarding, training, and managing employees from the time they apply for a job until they leave the company [10]. This comprehensive personnel management entails payroll and benefits administration, employee upskilling, developing a healthy workplace culture, improving staff productivity, managing employer–employee relationships, employee termination, and more. The human resources department is responsible for developing policies that create a balance between the interests of the company and the employees.

Today, the average human resources staff is responsible for more than just payroll and benefits administration, severance management, and post-retirement interactions. Their role now includes developing strategies to attract the ideal personnel, ensuring employee retention by resolving their concerns, managing employee separation, dealing with compliance and legal issues, and staying current on HR industry developments. Figure 9.1 shows some of the roles and responsibilities of the human resource department.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_9


**Fig. 9.1** Roles and responsibilities in human resource management [11]

Some of the tasks that human resources staff are responsible for include:

- **Recruitment**
  Human resources is in charge of filling open positions inside an organization. Creating and publishing job descriptions, receiving applications, shortlisting prospects, organizing interviews, hiring, and onboarding are all part of the recruiting process. Each of these recruitment processes includes planning the workflow and executing it in such a way that the top applicants are hired. Many teams rely on recruiting software to assist with the automation of these procedures.
- **Compensation and benefits**
  Handling employee salaries and benefits is a critical component of human resource duties. Compensation plays an essential role in recruiting and keeping top employees. HR must come up with inventive ways to compensate workers so that they are encouraged to remain and advance within the firm. In addition to payroll administration, it covers travel costs, paid vacation, sick leaves, retirement, health, and other benefits.
- **Talent management**
  Employee attrition is a serious concern for many companies, as losing valuable employees can be costly financially and time-consuming to replace. High attrition rates indicate a serious problem within the organization and can be caused by a variety of factors, from salary dissatisfaction to poor management

practices. In such cases, high attrition is a symptom of a bigger underlying issue that needs to be addressed. These issues are likely to harm the organization's brand, making it harder to attract talented individuals to join the company.

- **Defining organization culture**
  Human resources is responsible for more than recruiting employees and also ensuring that the environment is conducive to optimal performance. An organization needs a united culture to function effectively. HR must establish the desired corporate culture by creating workplace policies in cooperation with top management and ensuring that they are applied across the whole firm.
- **Compliance with labor laws**
  Human resource managers should also be familiar with the rules and regulations of the state in order to effectively manage the department and deal with any legal concerns that may surface in the workplace. Understanding the law will enable them to combat discrimination and abuse, avoid liabilities, and manage legal concerns in human resource management. Labor laws are enacted by the government to protect everyone, resolve disputes, and promote a healthy workplace. HR professionals must have a thorough understanding of the law and the ability to articulate it to both employers and workers. It is also the responsibility of the HR department to make sure that the business complies with and upholds labor and employment regulations.

## 9.2 Artificial Intelligence in Human Resources

The landscape of human resource management is always changing, and human resource managers have to adapt to the evolving workplace. For example, as economies become increasingly globalized and the trend of remote working has begun to take off, there has been an increasing shift to remote work. Organizations and employees both have to adapt to this shift and it is the human resources department to develop and execute policies that will redefine the future of work for the company. Artificial intelligence has the ability to assist human resource departments in helping human resource organizations automate repetitive tasks and provide insights to help drive decision-making in human resources. Here are some of the potential use cases that artificial intelligence can bring in human resource management:

1. **Hiring**
   Hiring for top talents is currently very competitive [3]. Candidates may already be hired by other competitors by the time you have reached out to them. In this aspect, artificial intelligence is able to speed up the recruitment process by crawling through a large number of applicants and ranking them based on their suitability for a role. Furthermore, screening software may help to eliminate unconscious biases by excluding features from personal information such as race, age, and gender. Chatbots can also help in the recruitment process by scheduling

meetings, answering potential questions, and providing important information about the interview process.

2. **Employee retention**
   After recruiting a great employee, the next challenge is to retain them. Artificial intelligence is able to help in this aspect by analyzing large amounts of data and providing insights to human resource departments that could help reduce attrition. AI can be used to help to ensure that employees are compensated fairly through analyzing salaries to keep up with the market rates. Additionally, AI can be used to predict the likelihood of attrition using data collected from salaries, surveys, and performance reviews. This can be used to help identify groups that are at risk of higher-than-normal turnover and help facilitate resource and policy planning.
3. **Human resource policy-making**
   Human resources performs an important role in developing policies that shape an organization's culture and workforce. Thus, it is important for them to be able to pay attention to the latest changes in the labor market. This can be done to analyze a large number of reviews to identify important features and rising trends, such as work–life balance or remote working, that can help attract or retain employees.
4. **Learning and development**
   Artificial intelligence can be utilized in learning and development to create personalized training recommendations, tailored to each staff. This can be achieved by developing recommender systems that rank and recommend courses that an employee might be interested in. Improving learning and development can help develop the organization's workforce and improve employee engagement.
5. **Reduce administrative workload**
   Artificial intelligence can be used to help automate various HR functions such as scheduling and facilitating leave requests. This can help reduce administrative friction, improve employee satisfaction, and free up precious time for human resource staff to perform other important tasks.

In conclusion, artificial intelligence has the potential to change and shape the future of human resources through various applications, from streamlining the recruitment process to developing individualized learning programs for employees. This can help improve the overall efficiency of organizations and provide employees with a better working experience. Now, we will go through some simple examples of how artificial intelligence can be applied in human resource management.

## 9.3 Applications of AI in Human Resources

### *9.3.1 Salary Prediction*

Ensuring that employees are well compensated is important for improving the overall morale and motivation of the staff, as well as reducing turnover, which is

vital to the long-term success of any business. Companies that pay their employees well are in a much better position to attract and retain talented staff, which is key to establishing a successful business. In this respect, artificial intelligence can be used to predict worker salaries based on their job descriptions and skills. This can be used to provide better salary offers to potential employees and ensure that existing employees are properly compensated for their work. Companies that are able to use artificial intelligence to accurately predict worker salaries will be better positioned to attract and retain the best talent and, in turn, build a more successful business.

In this example, we will develop a model to predict the salaries of data scientists using data scrapped from glassdoor [5]. We can access the dataset from:

https://www.kaggle.com/datasets/lokkagle/glassdoor-data

This dataset contains various job titles, descriptions, ratings, companies, and salaries of data scientists across the United States.

Now, let us begin by importing the relevant packages

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
```

Let us take a better look at the data.

```
# loading the data
data = pd.read_csv('Human_Resources/glassdoor_jobs.csv
↪')
data.head(3)
```

```
   Unnamed: 0                  Job Title              Salary Estimate  \
0           0             Data Scientist   $53K-$91K (Glassdoor est.)
1           1  Healthcare Data Scientist  $63K-$112K (Glassdoor est.)
2           2             Data Scientist   $80K-$90K (Glassdoor est.)

                                  Job Description  Rating  \
0  Data Scientist\nLocation: Albuquerque, NM\nEdu...     3.8
1  What You Will Do:\n\nI. General Summary\n\nThe...     3.4
2  KnowBe4, Inc. is a high growth information sec...     4.8

                              Company Name         Location  \
0                    Tecolote Research\n3.8  Albuquerque, NM
1  University of Maryland Medical System\n3.4    Linthicum, MD
2                              KnowBe4\n4.8   Clearwater, FL

     Headquarters                   Size  Founded   Type of ownership  \
0      Goleta, CA  501 to 1000 employees     1973   Company - Private
```

(continues on next page)

(continued from previous page)

```
1   Baltimore, MD       10000+ employees     1984  Other Organization
2  Clearwater, FL  501 to 1000 employees     2010   Company - Private

                          Industry               Sector  \
0               Aerospace & Defense  Aerospace & Defense
1  Health Care Services & Hospitals          Health Care
2                 Security Services    Business Services

                     Revenue Competitors
0   $50 to $100 million (USD)          -1
1      $2 to $5 billion (USD)          -1
2  $100 to $500 million (USD)          -1
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 956 entries, 0 to 955
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          956 non-null    int64
 1   Job Title           956 non-null    object
 2   Salary Estimate     956 non-null    object
 3   Job Description     956 non-null    object
 4   Rating              956 non-null    float64
 5   Company Name        956 non-null    object
 6   Location            956 non-null    object
 7   Headquarters        956 non-null    object
 8   Size                956 non-null    object
 9   Founded             956 non-null    int64
 10  Type of ownership   956 non-null    object
 11  Industry            956 non-null    object
 12  Sector              956 non-null    object
 13  Revenue             956 non-null    object
 14  Competitors         956 non-null    object
dtypes: float64(1), int64(2), object(12)
memory usage: 112.2+ KB
```

```
data.isna().sum()
```

```
Unnamed: 0           0
Job Title            0
Salary Estimate      0
Job Description      0
```

(continues on next page)

(continued from previous page)

```
Rating                0
Company Name          0
Location              0
Headquarters          0
Size                  0
Founded               0
Type of ownership     0
Industry              0
Sector                0
Revenue               0
Competitors           0
dtype: int64
```

Now, let us copy the data and remove the "Unnamed: 0" column which provides no useful information.

```
# take a copy of data and remove unnecessary␣
↪attributes
emp_data = data.copy(deep= True)
emp_data.drop(columns= ['Unnamed: 0'], inplace = True)
emp_data.head()
```

```
                  Job Title              Salary Estimate  \
0            Data Scientist   $53K-$91K (Glassdoor est.)
1  Healthcare Data Scientist  $63K-$112K (Glassdoor est.)
2            Data Scientist   $80K-$90K (Glassdoor est.)
3            Data Scientist   $56K-$97K (Glassdoor est.)
4            Data Scientist  $86K-$143K (Glassdoor est.)

                                     Job Description  Rating  \
0  Data Scientist\nLocation: Albuquerque, NM\nEdu...     3.8
1  What You Will Do:\n\nI. General Summary\n\nThe...     3.4
2  KnowBe4, Inc. is a high growth information sec...     4.8
3  *Organization and Job ID**\nJob ID: 310709\n\n...     3.8
4  Data Scientist\nAffinity Solutions / Marketing...     2.9

                              Company Name         Location  \
0                    Tecolote Research\n3.8  Albuquerque, NM
1  University of Maryland Medical System\n3.4    Linthicum, MD
2                             KnowBe4\n4.8   Clearwater, FL
3                                PNNL\n3.8     Richland, WA
4                  Affinity Solutions\n2.9     New York, NY

     Headquarters                    Size  Founded   Type of ownership  \
0      Goleta, CA   501 to 1000 employees     1973   Company - Private
1   Baltimore, MD        10000+ employees     1984  Other Organization
2  Clearwater, FL   501 to 1000 employees     2010   Company - Private
3    Richland, WA  1001 to 5000 employees     1965          Government
4    New York, NY     51 to 200 employees     1998   Company - Private
```

(continues on next page)

(continued from previous page)

```
                          Industry                        Sector  \
0              Aerospace & Defense          Aerospace & Defense
1  Health Care Services & Hospitals                Health Care
2                Security Services            Business Services
3                           Energy  Oil, Gas, Energy & Utilities
4         Advertising & Marketing             Business Services

                          Revenue  \
0        $50 to $100 million (USD)
1           $2 to $5 billion (USD)
2       $100 to $500 million (USD)
3  $500 million to $1 billion (USD)
4          Unknown / Non-Applicable

                                        Competitors
0                                                -1
1                                                -1
2                                                -1
3  Oak Ridge National Laboratory, National Renewa...
4              Commerce Signals, Cardlytics, Yodlee
```

Next, there are many variations in the job titles, and let us clean it up by categorizing them into their relevant categories.

```
# job title cleaning

def jobtitle_cleaner(title):
    if 'data scientist' in title.lower():
        return 'D-sci'
    elif 'data engineer' in title.lower():
        return 'D-eng'
    elif 'analyst' in title.lower():
        return 'analyst'
    elif 'machine learning' in title.lower():
        return 'ML'
    elif 'manager' in title.lower():
        return 'manager'
    elif 'director' in title.lower():
        return 'director'
    elif 'research' in title.lower():
        return 'R&D'
    else:
        return 'na'

emp_data['JobTitles'] = emp_data['Job Title'].
↪apply(jobtitle_cleaner)
emp_data['JobTitles'].value_counts()
```

```
D-sci        358
na           219
D-eng        158
analyst      124
manager       36
ML            26
R&D           19
director      16
Name: JobTitles, dtype: int64
```

Now, we will need to split the categories into senior, junior, and other roles.

```
senior_list = ['sr','sr.','senior','principal',
↪'research', 'lead', 'R&D','II', 'III']
junior_list = ['jr','jr.','junior']


def jobseniority(title):
    for i in senior_list:
        if i in title.lower():
            return 'Senior'

    for j in junior_list:
        if j in title.lower():
            return 'Junior'
        else:
            return 'No Desc'

emp_data['Job Seniority'] = emp_data['Job Title'].
↪apply(jobseniority)
emp_data['Job Seniority'].value_counts()
```

```
No Desc    671
Senior     283
Junior       2
Name: Job Seniority, dtype: int64
```

We will also be able to binarize their skill sets based on the skills specified in their job descriptions.

```
# job descriptions
jobs_list = ['python', 'excel','r studio', 'spark',
↪'aws']
```

(continues on next page)

(continued from previous page)

```
for i in jobs_list:
    emp_data[i+'_'+'job'] = emp_data['Job Description
↪'].apply(lambda x : 1 if i in x.lower() else 0)
for i in jobs_list:
    print(emp_data[i+'_'+'job'].value_counts())
```

```
1    496
0    460
Name: python_job, dtype: int64
1    486
0    470
Name: excel_job, dtype: int64
0    955
1      1
Name: r studio_job, dtype: int64
0    742
1    214
Name: spark_job, dtype: int64
0    714
1    242
Name: aws_job, dtype: int64
```

Now, let us clean up the company names.

```
emp_data['Company Name'] = emp_data['Company Name'].
↪apply(lambda x : x.split("\n")[0])
emp_data['Company Name'].value_counts()
```

```
Novartis                              14
MassMutual                            14
Takeda Pharmaceuticals                14
Reynolds American                     14
Software Engineering Institute        13
                                      ..
Systems Evolution Inc.                 1
Centro                                 1
comScore                               1
Genesis Research                       1
Fivestars                              1
Name: Company Name, Length: 448, dtype: int64
```

There are many cities that these data scientists are working at and the counts would be too small to extract anything meaningful. Let us aggregate the data by grouping the cities into states instead.

```
emp_data['location spots'] = emp_data['Location'].str.
↪split(',').str[1]
emp_data['location spots'].value_counts().head()
```

```
 CA    210
 MA    124
 NY     96
 VA     56
 IL     48
Name: location spots, dtype: int64
```

Now we will clean up the companies of competitors by specifying only the top competitor.

```
emp_data['competitor company'] = emp_data['Competitors
↪'].str.split(',').str[0].replace('-1', 'no␣
↪competitors')
emp_data['competitor company'].value_counts()
```

```
no competitors                    634
Novartis                           14
Oak Ridge National Laboratory      12
Travelers                          11
Roche                               9
                                 ...
Greystar                            1
Ecolab                              1
USAA                                1
Clearlink                           1
Belly                               1
Name: competitor company, Length: 137, dtype: int64
```

Clean up the ownership column as well.

```
emp_data['Ownership'] = emp_data['Type of ownership'].
↪str.split('-').str[1].replace(np.NaN, 'others')
emp_data['Ownership'].value_counts()
```

```
 Private    532
 Public     237
others      176
1            11
Name: Ownership, dtype: int64
```

We can also specify the revenue of the company that these data scientists work for to get a better estimate.

```
emp_data['Revenue'] = emp_data['Revenue'].str.replace(
↪'-1','others')
```

Define the Headquarter location by the state instead of city as well.

```
emp_data['HQ'] = emp_data['Headquarters'].str.split(',
↪').str[1]
```

Clean up the company size.

```
emp_data['Size'] = emp_data['Size'].str.replace('-1',
↪'others')
emp_data['Size'].value_counts()
```

```
1001 to 5000 employees     177
201 to 500 employees       160
51 to 200 employees        155
10000+ employees           154
501 to 1000 employees      144
5001 to 10000 employees     79
1 to 50 employees           61
Unknown                     15
others                      11
Name: Size, dtype: int64
```

Now we will need to extract the minimum, maximum, and average salary from the salary range data provided by glassdoor. This requires cleaning up the text data and filling up the empty rows with the mean value.

```
import warnings
warnings.simplefilter(action='ignore',␣
↪category=FutureWarning)
emp_data['min_sal'] = emp_data['Salary Estimate'].str.
↪split("(").str[0].str.replace('(Glassdoor est.)','')
emp_data['min_sal'] = emp_data['min_sal'].str.replace(
↪'(Glassdoor est.)','').str.split('-').str[0].str.
↪replace('$','')
emp_data['min_sal'] = emp_data['min_sal'].str.replace(
↪'K','')
emp_data['min_sal'] = emp_data['min_sal'].str.replace(
↪'Employer Provided Salary:','')
```

(continues on next page)

(continued from previous page)

```
emp_data['min_sal'] = pd.to_numeric(emp_data['min_sal
↪'], errors='coerce')
emp_data['min_sal'] = emp_data['min_sal'].replace(np.
↪nan, emp_data['min_sal'].mean())

emp_data['max_sal'] = emp_data['Salary Estimate'].str.
↪split("(").str[0].str.replace('(Glassdoor est.)','')
emp_data['max_sal'] = emp_data['max_sal'].str.replace(
↪'(Glassdoor est.)','').str.split('-').str[0].str.
↪replace('$','')
emp_data['max_sal'] = emp_data['max_sal'].str.replace(
↪'K','')
emp_data['max_sal'] = emp_data['max_sal'].str.replace(
↪'Employer Provided Salary:','')
emp_data['max_sal'] = pd.to_numeric(emp_data['max_sal
↪'], errors='coerce')
emp_data['max_sal'] = emp_data['max_sal'].replace(np.
↪nan, emp_data['min_sal'].mean())

emp_data['avg.salary'] = (emp_data['min_sal'] + emp_
↪data['max_sal'])/ 2
```

Now that the text is all cleaned up, let us create dummy variables to represent our categorical data columns.

```
processed_data = emp_data[['Rating',
       'Company Name', 'Size',
       'Type of ownership','Sector', 'Revenue',
       'JobTitles', 'Job Seniority', 'python_job',
↪'excel_job', 'r studio_job',
       'spark_job', 'aws_job', 'HQ', 'location spots',
       'competitor company', 'Ownership','avg.salary
↪']]
processed_data = pd.get_dummies(data = processed_data,
↪ columns = ['Company Name', 'Size', 'Type of␣
↪ownership', 'Sector',
       'Revenue', 'JobTitles', 'Job Seniority','HQ',
↪'location spots',
       'competitor company', 'Ownership'])
processed_data.head()
```

```
   Rating  python_job  excel_job  r studio_job  spark_job  aws_job  \
0     3.8           1          1             0          0        0
1     3.4           1          0             0          0        0
```

(continues on next page)

(continued from previous page)

```
2     4.8           1          1             0          1        0
3     3.8           1          0             0          0        0
4     2.9           1          1             0          0        0

   avg.salary  Company Name_1-800-FLOWERS.COM, Inc.  Company Name_1904labs  \
0        53.0                                    0                      0
1        63.0                                    0                      0
2        80.0                                    0                      0
3        56.0                                    0                      0
4        86.0                                    0                      0

   Company Name_23andMe  ...  competitor company_World Wide Technology  \
0                     0  ...                                         0
1                     0  ...                                         0
2                     0  ...                                         0
3                     0  ...                                         0
4                     0  ...                                         0

   competitor company_YOOX NET-A-PORTER GROUP  competitor company_Zocdoc  \
0                                           0                          0
1                                           0                          0
2                                           0                          0
3                                           0                          0
4                                           0                          0

   competitor company_bluebird bio  competitor company_eClinicalWorks  \
0                                 0                                  0
1                                 0                                  0
2                                 0                                  0
3                                 0                                  0
4                                 0                                  0

   competitor company_no competitors  Ownership_ Private  Ownership_ Public  \
0                                  1                   1                  0
1                                  1                   0                  0
2                                  1                   1                  0
3                                  0                   0                  0
4                                  0                   1                  0

   Ownership_1  Ownership_others
0            0                0
1            0                1
2            0                0
3            0                1
4            0                0

[5 rows x 758 columns]
```

We will apply the min–max scaler to scale all features to between 0 and 1 so that all features are equally weighted by magnitude and will not skew the data with extreme numbers.

```
ms = MinMaxScaler()
scaled_data = processed_data.copy()
scaled_data[['Rating', 'avg.salary']] = ms.fit_
↪transform(processed_data[['Rating', 'avg.salary']])

X = scaled_data.drop(columns= 'avg.salary').values
y = scaled_data["avg.salary"].values
```

Now that we are done, let us split the data and train our model.

```
X_train, X_test, y_train, y_test = train_test_split(X,
 ↪ y, test_size = 0.2, random_state = 0)
# Creating random forest regression model
model = RandomForestRegressor(n_estimators=100,␣
 ↪criterion='mse', random_state=0)
# Fitting the dataset to the model
model.fit(X_train, y_train)

min_avg_salary = processed_data["avg.salary"].min()
max_avg_salary = processed_data["avg.salary"].max()

train_error = mean_absolute_error(y_train, model.
 ↪predict(X_train))
train_error = (train_error * (max_avg_salary - min_
 ↪avg_salary) * 1000)
print("Average training error: ${0}".
 ↪format(round(train_error, 2)))
test_error = mean_absolute_error(y_test, model.
 ↪predict(X_test))
test_error = test_error * (max_avg_salary - min_avg_
 ↪salary) * 1000
print("Average test error: ${0}".format(round(test_
 ↪error, 2)))
```

```
Average training error: $3799.53
Average test error: $9666.36
```

Now we have a model that can estimate salaries with an error of approximately $9.7k in annual compensation. This can be further improved with more data and finer grained features extracted from the job description and titles.

### *9.3.2 Recruitment*

One of the most challenging parts of recruitment is the process of screening candidates from a long list of applications. It is very time-consuming and tedious. An average recruiter would spend approximately 15 hours to identify a suitable hire [6]. This problem is exacerbated by the ease of job searching in the digital age. As thousands of jobs can be easily found through a quick web search, the volumes of applications continue to increase significantly. With the same amount of manpower, human resource staff have to become more efficient in screening through resumes. This is where artificial intelligence can shine in the recruitment process as it can

help automate and sort through a large number of resumes to provide rankings on the suitability of candidates. Furthermore, unconscious human biases can be reduced through this initial screening process. In this example, we will develop a simple AI algorithm to demonstrate how it can be used in the recruitment process [9].

We will be using the data provided from this link [8]:
https://www.kaggle.com/datasets/rafunlearnhub/recruitment-data

The dataset contains features of various candidates for a recruitment process. The outcome is labeled under the Recruitment_Status column which indicates whether a candidate was eventually recruited or not.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_
↪matrix
```

We will import and take a quick look at our data.

```
df = pd.read_csv('Human_Resources/recruitment_
↪decision_tree.csv')
df.head()
```

| | Serial_no | Gender | Python_exp | Experience_Years | Education | Internship | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | Male | Yes | 0.0 | Graduate | No | |
| 1 | 2 | Male | No | 1.0 | Graduate | No | |
| 2 | 3 | Male | No | 0.0 | Graduate | Yes | |
| 3 | 4 | Male | No | 0.0 | Not Graduate | No | |
| 4 | 5 | Male | Yes | 0.0 | Graduate | No | |

| | Score | Salary * 10E4 | Offer_History | Location | Recruitment_Status |
|---|---|---|---|---|---|
| 0 | 5139 | 0.0 | 1.0 | Urban | Y |
| 1 | 4583 | 128.0 | 1.0 | Rural | N |
| 2 | 3000 | 66.0 | 1.0 | Urban | Y |
| 3 | 2583 | 120.0 | 1.0 | Urban | Y |
| 4 | 6000 | 141.0 | 1.0 | Urban | Y |

Okay, our data contain various categorical and Boolean variables. Let us map them into numbers.

```
sex_map = {'Male':1, 'Female':0}
df['Gender'] = df['Gender'].map(sex_map)

pyth_map = {'Yes':1, 'No':0}
df['Python_exp'] = df['Python_exp'].map(pyth_map)
```

(continues on next page)

(continued from previous page)

```
edu_map = {'Graduate':1, 'Not Graduate':0}
df['Education'] = df['Education'].map(edu_map)

intern_map = {'Yes':1, 'No':0}
df['Internship'] = df['Internship'].map(intern_map)

locat_map = {'Urban':3, 'Semiurban':2, 'Rural':1}
df['Location'] = df['Location'].map(locat_map)

status_map = {'Y':1, 'N':0}
df['Recruitment_Status'] = df['Recruitment_Status'].
↪map(status_map)

df = df.drop('Serial_no', axis=1)
df.head()
```

```
   Gender  Python_exp  Experience_Years  Education  Internship  Score  \
0     1.0         1.0               0.0          1         0.0   5139
1     1.0         0.0               1.0          1         0.0   4583
2     1.0         0.0               0.0          1         1.0   3000
3     1.0         0.0               0.0          0         0.0   2583
4     1.0         1.0               0.0          1         0.0   6000

   Salary * 10E4  Offer_History  Location  Recruitment_Status
0            0.0            1.0         3                   1
1          128.0            1.0         1                   0
2           66.0            1.0         3                   1
3          120.0            1.0         3                   1
4          141.0            1.0         3                   1
```

Now, let us check for empty cells.

```
df.isna().sum()
```

```
Gender                13
Python_exp             3
Experience_Years      15
Education              0
Internship            32
Score                  0
Salary * 10E4         21
Offer_History         50
Location               0
Recruitment_Status     0
dtype: int64
```

We will set female as the default gender, assume that blanks have no previous offers or internships, set the years of job experience and Python experience to be 0, and set blanks in salary to be the mean value.

```
df['Gender'].fillna(0, inplace=True)

df['Experience_Years'].fillna(0, inplace=True)

df['Python_exp'].fillna(0, inplace=True)

df['Offer_History'].fillna(0, inplace=True)

df['Internship'].fillna(0, inplace=True)

df['Salary * 10E4'].fillna(np.mean(df['Salary * 10E4
↪']), inplace=True)

df.describe()
```

```
           Gender  Python_exp  Experience_Years   Education  Internship  \
count  614.000000  614.000000        614.000000  614.000000  614.000000
mean     0.796417    0.346906          0.744300    0.781759    0.133550
std      0.402991    0.476373          1.009623    0.413389    0.340446
min      0.000000    0.000000          0.000000    0.000000    0.000000
25%      1.000000    0.000000          0.000000    1.000000    0.000000
50%      1.000000    0.000000          0.000000    1.000000    0.000000
75%      1.000000    1.000000          1.000000    1.000000    0.000000
max      1.000000    1.000000          3.000000    1.000000    1.000000

              Score  Salary * 10E4  Offer_History    Location  \
count    614.000000     614.000000     614.000000  614.000000
mean    5402.302932     146.165261       0.773616    2.037459
std     6109.024398      84.244922       0.418832    0.787482
min      150.000000       0.000000       0.000000    1.000000
25%     2877.500000     100.000000       1.000000    1.000000
50%     3812.500000     128.500000       1.000000    2.000000
75%     5771.500000     164.750000       1.000000    3.000000
max    81000.000000     700.000000       1.000000    3.000000

       Recruitment_Status
count          614.000000
mean             0.687296
std              0.463973
min              0.000000
25%              0.000000
50%              1.000000
75%              1.000000
max              1.000000
```

We will need to split our data into training and testing accordingly. In this case we will place 30% of our data into the testing set.

```
np.random.seed(42)
idx = np.random.rand(len(df))<0.70

train = df[idx]
test = df[~idx]

x_train = train.drop('Recruitment_Status', axis=1)
y_train = train[['Recruitment_Status']]

x_test = test.drop('Recruitment_Status', axis=1)
y_test = test[['Recruitment_Status']]
```

As our dataset is imbalanced in terms of classes, we will apply Synthetic Minority Oversampling Technique (SMOTE) to synthesize more of the minority class by interpolating between the features of the existing data points across the minority class.

```
oversample = SMOTE(random_state=0)
x_train, y_train = oversample.fit_resample(x_train, y_
↪train)
```

Now we can train our model and predict on our test set.

```
model = DecisionTreeClassifier(criterion="entropy",␣
↪max_depth=4, random_state=0)
model.fit(x_train, y_train)

y_test_pred = model.predict(x_test)
```

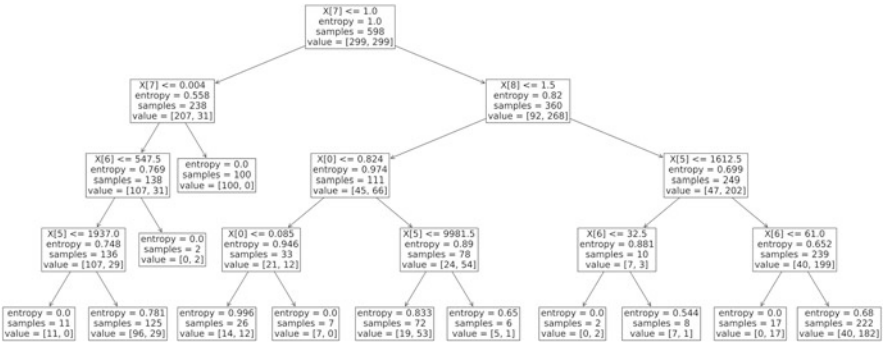
Let us see how our model does.

```
print(confusion_matrix(y_test, y_test_pred))
print(accuracy_score(y_test, y_test_pred))
```

```
[[ 31  33]
 [ 17 106]]
0.732620320855615
```

Now we have created a model that can predict suitable candidates with 73% accuracy. Let us visualize the tree.

```
plt.figure(0, figsize=(35, 15))
tree.plot_tree(model, fontsize=20)
plt.show()
```




```
df.columns[8]
```

```
'Location'
```

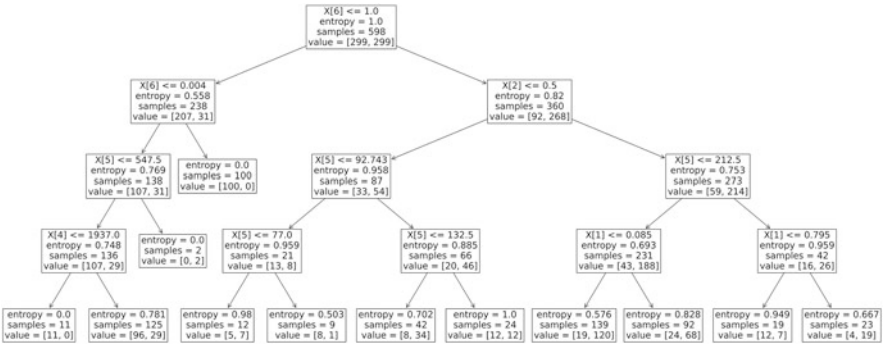
As you can see, the model places significant weightage on Location (x[7]) and Gender (x[0]) which may not be ideal. Gender bias can be removed by simply excluding such features from the model and location may not matter as much if companies are willing to spend more to support recruiting top talent around the world. Let us see what happens when we remove these features.

```
x_train2 = x_train.drop(['Gender', 'Location'],␣
↪axis=1)
x_test2 = x_test.drop(['Gender', 'Location'], axis=1)

model = DecisionTreeClassifier(criterion="entropy",␣
↪max_depth=4, random_state=0)
model.fit(x_train2, y_train)

y_test_pred = model.predict(x_test2)

print(confusion_matrix(y_test, y_test_pred))
print(accuracy_score(y_test, y_test_pred))
```

```
[[ 34  30]
 [ 21 102]]
0.7272727272727273
```

It looks like our model seems to perform pretty well even without using these features. Let us visualize how it decides who is worth recruiting.

```
plt.figure(0, figsize=(35, 15))
tree.plot_tree(model, fontsize=20)
plt.show()
```




Let us use a more complex model such as a Random Forest which is an ensemble of decision trees generated by randomly sampling subsets of the training set with replacement. The classes are determined by aggregating the votes across all decision trees.

```
model = RandomForestClassifier(random_state=0)
model.fit(x_train2, np.ravel(y_train))

y_test_pred = model.predict(x_test2)

print(confusion_matrix(y_test, y_test_pred))
print(accuracy_score(y_test, y_test_pred))
```

```
[[ 31  33]
 [ 14 109]]
0.7486631016042781
```

We are able to get up to 74.8% accuracy with the random forest. Now we can rank our candidates using our model.

```
probs = model.predict_proba(x_test2)[:, 1]
most_suitable_candidates = probs.argsort()[::-1][0:10]
print(most_suitable_candidates, probs[most_suitable_
↪candidates])
```

```
[ 90  37 174  11  68  24  91 113  22  47] [1.   1.   1.   0.99 0.99 0.99 0.
↪99 0.99 0.98 0.98]
```

As you can see, the model is able to recommend the top few candidates that it believes suitable. This is a simple model that demonstrates what can be done with categorical and numerical variables. More complex models can be developed with more data and more descriptive features. For example, sentences in resumes can be processed by natural language processing and suitability can be assessed based on how well their resume matches a query job description to further improve the relevance to each individual job application.

### *9.3.3 Course Recommendation*

Professionals require training to fill up their knowledge gaps, required for both their present responsibilities and career progression. Training and development play a significant role in fostering employee confidence and happiness at work. Thus, it is essential for HR departments to recognize skill gaps among employees and conduct training programs for them.

Training needs could originate from the employee, the employee's supervisor, or job-specific requirements. Identifying such needs requires the collection and consolidation of data. Looking at training programs that are popular among peers with similar interests is one of the ways to go about it. AI will be able to assist by analyzing evaluations for various training programs and identifying programs that a particular employee may enjoy. In this example, we will look into how AI can be used for course recommendation [2].

Collaborative filtering is one of the techniques applied in recommendation systems for user item suggestion. This approach detects comparable users based on the items they share in common. For each user, the system suggests goods that comparable users have liked or purchased. Numerous e-commerce websites utilize this method to produce product suggestions for visitors. In the exercise, we will demonstrate how we can modify the process to develop a system for course recommendations.

We will be using the employee course ratings dataset from this link [1]:

https://www.kaggle.com/datasets/aryashah2k/datasets-in-hr-analytics-applied-ai?select=employee_course_ratings.csv

The dataset consists of employee evaluations collected from courses they completed as part of their training programs. The dataset consists of the employee ID, name of the employee who completed the course, course ID, course name, and the associated rating provided at the end of the course. The rating scores range between one and five, with five being the highest.

We will develop an algorithm to predict an employee's rating of a course that he or she has not taken yet. We expect that employees would submit a higher rating if they found the training beneficial.

Let us begin by importing the libraries and dataset.

```
from csv import reader
import pandas as pd
import os
import numpy as np
import math
import torch
from sklearn.model_selection import train_test_split

ratings_data = pd.read_csv("Human_Resources/employee_
↪course_ratings.csv")

ratings_data.head()
```

| | EmployeeID | EmpName | CourseID | CourseName | Rating |
|---|---|---|---|---|---|
| 0 | 1408 | Ignace Ormonde | 14 | Video Production | 3 |
| 1 | 1249 | Gabriela Balcon | 17 | Translation | 2 |
| 2 | 1158 | Enrique Lewer | 8 | IT Architecture | 3 |
| 3 | 1564 | Wallie Byrd | 18 | Natural Language Processing | 3 |
| 4 | 1334 | Hannah Ganter | 6 | Java Programming | 4 |

We will build a list of unique employees and courses.

```
#Build list of unique Employees
emp_list=ratings_data.groupby(
    ['EmployeeID','EmpName']).size().reset_index()
emp_list.head()
print("Total Employees: ",len(emp_list))

#Build list of unique Courses
course_list=ratings_data.groupby(
    ['CourseID','CourseName']).size().reset_index()
course_list.head()
print("Total Courses: ", len(course_list))
```

```
Total Employees:  638
Total Courses:  25
```

Now we will need to split the data into training, validation, and testing.

```
ratings_train_val, ratings_test = train_test_
↪split(ratings_data, test_size=0.1)
ratings_train, ratings_val = train_test_split(ratings_
↪train_val, test_size=0.1)
```

We will need to define a torch dataset and dataloader in order to load our data for training and evaluation.

```
class ReviewDataset(torch.utils.data.Dataset):
    def __init__(self, df):
        self.df = df

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        feats = torch.tensor([row["EmployeeID"], row[
↪"CourseID"]])
        labels = torch.tensor(row["Rating"], dtype =␣
↪torch.FloatTensor.dtype)
        return feats, labels

    def __len__(self):
        return len(self.df)

train_dataset = ReviewDataset(ratings_train)
val_dataset = ReviewDataset(ratings_val)
test_dataset = ReviewDataset(ratings_test)
train_dataloader = torch.utils.data.DataLoader(train_
↪dataset, batch_size=64, shuffle=True)
val_dataloader = torch.utils.data.DataLoader(val_
↪dataset, batch_size=64, shuffle=True)
test_dataloader = torch.utils.data.DataLoader(test_
↪dataset, batch_size=64, shuffle=True)
```

Word embedding is a common language modeling and feature learning technique often applied in deep learning, especially in the field of natural language processing. Word embeddings translate text into numerical values and represent the connections between words using vectors. After training on a dataset, it captures how various words relate to one another, with co-occurring terms receiving greater similarity scores.

In this use case, we will utilize employee IDs and course IDs instead of words to construct embeddings. Each employee ID and course ID will be represented by its own embedding. For each data point, we will concatenate the course ID embedding with the employee ID embedding before passing it through our network.

```
class CourseRecommender(torch.nn.Module):
    def __init__(self, emp_emb_size = 5, course_emb_
↪size = 5):
        super(CourseRecommender, self).__init__()
        self.employee_embedding = torch.nn.
↪Embedding(max(emp_list), emp_emb_size)
```

(continues on next page)

(continued from previous page)

```
        self.course_embedding = torch.nn.
↪Embedding(len(course_list)+1, course_emb_size)
        self.fc1 = torch.nn.Linear(emp_emb_size +␣
↪course_emb_size, 128)
        self.relu1 = torch.nn.ReLU()
        self.fc2 = torch.nn.Linear(128, 32)
        self.relu2 = torch.nn.ReLU()
        self.fc3 = torch.nn.Linear(32, 1)

    def forward(self, embedding):
        emp_x = self.employee_embedding(embedding[:,␣
↪0])
        course_x = self.course_embedding(embedding[:,␣
↪1])
        merged = torch.cat((emp_x, course_x), dim = 1)
        x = self.fc1(merged)
        x = self.relu1(x)
        x = self.fc2(x)
        x = self.relu2(x)
        x = self.fc3(x)
        x = x.squeeze(1)
        return x
```

Now that we have set up our model and dataset, let us define the loss and optimizer and begin training the model.

```
model = CourseRecommender()
criterion = torch.nn.MSELoss()
optimizer = torch.optim.NAdam(model.parameters(),␣
↪lr=0.001)

best_loss = np.inf
for epoch in range(50):
    training_loss = []
    validation_loss = []

    for feats, labels in train_dataloader:
        optimizer.zero_grad()
        loss = criterion(model(feats), labels)
        loss.backward()
        optimizer.step()
```

(continues on next page)

(continued from previous page)

```
        training_loss.append(loss.detach().cpu().
↪item())

    with torch.no_grad():
        for feats, labels in val_dataloader:
            loss = criterion(model(feats), labels)
            validation_loss.append(loss.detach().
↪cpu().item())

    training_loss = np.array(training_loss).mean()
    validation_loss = np.array(validation_loss).mean()

    if validation_loss < best_loss:
        best_model_statedict = model.state_dict()
        best_loss = validation_loss

    print("Epoch: {0}, Training Loss: {1}, Validation␣
↪Loss: {2}".format(epoch,

↪                  np.array(training_loss).mean(),

↪                  np.array(validation_loss).mean()

↪                  ))
```

```
Epoch: 0, Training Loss: 7.713223970853365, Validation Loss: 6.383551836013794
Epoch: 1, Training Loss: 4.214089925472553, Validation Loss: 2.992550849914551
Epoch: 2, Training Loss: 2.1887883223020115, Validation Loss: 2.358051896095276
Epoch: 3, Training Loss: 1.9668291165278509, Validation Loss: 2.4290822744369507
Epoch: 4, Training Loss: 1.9138606878427358, Validation Loss: 2.4183861017227173
Epoch: 5, Training Loss: 1.867754074243399, Validation Loss: 2.1712432503700256
Epoch: 6, Training Loss: 1.8279758049891546, Validation Loss: 2.144009292125702
Epoch: 7, Training Loss: 1.8098880419364343, Validation Loss: 2.1635595560073853
Epoch: 8, Training Loss: 1.7635837426552405, Validation Loss: 2.004770040512085
Epoch: 9, Training Loss: 1.7478572313602154, Validation Loss: 1.9925899505615234
Epoch: 10, Training Loss: 1.7204478337214544, Validation Loss: 2.3151422142982483
Epoch: 11, Training Loss: 1.7139681027485774, Validation Loss: 2.2434771060943604
Epoch: 12, Training Loss: 1.6844779253005981, Validation Loss: 2.1030672788619995
Epoch: 13, Training Loss: 1.665182507955111, Validation Loss: 2.169582962989807
Epoch: 14, Training Loss: 1.6464892350710356, Validation Loss: 2.3010743856430054
Epoch: 15, Training Loss: 1.621514888910147, Validation Loss: 2.233031749725342
Epoch: 16, Training Loss: 1.6048019665938158, Validation Loss: 2.1360620260238647
Epoch: 17, Training Loss: 1.5979149433282704, Validation Loss: 2.0288277864456177
Epoch: 18, Training Loss: 1.5910156139960656, Validation Loss: 1.991523027420044
Epoch: 19, Training Loss: 1.574485448690561, Validation Loss: 2.053059697151184
Epoch: 20, Training Loss: 1.559514696781452, Validation Loss: 2.073699951171875
Epoch: 21, Training Loss: 1.540119693829463, Validation Loss: 2.172857403755188
```

(continues on next page)

(continued from previous page)

```
Epoch: 22, Training Loss: 1.522446687404926, Validation Loss: 2.1322072744369507
Epoch: 23, Training Loss: 1.5124560319460356, Validation Loss: 2.1579567193984985
Epoch: 24, Training Loss: 1.4868639065669134, Validation Loss: 2.0146883726119995
Epoch: 25, Training Loss: 1.481570372214684, Validation Loss: 1.9257043600082397
Epoch: 26, Training Loss: 1.4848324427237878, Validation Loss: 1.9063563346862793
Epoch: 27, Training Loss: 1.453480234512916, Validation Loss: 2.0531848073005676
Epoch: 28, Training Loss: 1.4430676698684692, Validation Loss: 2.1641669869422913
Epoch: 29, Training Loss: 1.4371415835160475, Validation Loss: 2.013516664505005
Epoch: 30, Training Loss: 1.423480198933528, Validation Loss: 1.915759265422821
Epoch: 31, Training Loss: 1.4072460578038142, Validation Loss: 2.134285807609558
Epoch: 32, Training Loss: 1.3975641268950243, Validation Loss: 2.0174754858016968
Epoch: 33, Training Loss: 1.3734032190763032, Validation Loss: 1.9743831157684326
Epoch: 34, Training Loss: 1.3530409473639269, Validation Loss: 1.9249634146690369
Epoch: 35, Training Loss: 1.3571633192209096, Validation Loss: 1.9951313734054565
Epoch: 36, Training Loss: 1.3370022498644316, Validation Loss: 1.6750591397285461
Epoch: 37, Training Loss: 1.3319801550645094, Validation Loss: 2.0654801726341248
Epoch: 38, Training Loss: 1.2990752091774573, Validation Loss: 1.8388822674751282
Epoch: 39, Training Loss: 1.2814254027146559, Validation Loss: 1.896884262561798
Epoch: 40, Training Loss: 1.277358889579773, Validation Loss: 1.8458614945411682
Epoch: 41, Training Loss: 1.2613131541472216, Validation Loss: 1.7650485634803772
Epoch: 42, Training Loss: 1.236700796163999, Validation Loss: 2.0034565329551697
Epoch: 43, Training Loss: 1.2196704332645123, Validation Loss: 1.8137983083724976
Epoch: 44, Training Loss: 1.2169052912638738, Validation Loss: 1.90548837184906
Epoch: 45, Training Loss: 1.1830403483830965, Validation Loss: 1.9475241303443909
Epoch: 46, Training Loss: 1.1710590766026423, Validation Loss: 2.053204596042633
Epoch: 47, Training Loss: 1.1692262979654164, Validation Loss: 1.7238733768463135
Epoch: 48, Training Loss: 1.1471708921285777, Validation Loss: 1.804313838481903
Epoch: 49, Training Loss: 1.1309779653182397, Validation Loss: 2.1029959321022034
```

Now, let us evaluate our model on the test set.

```
model.load_state_dict(best_model_statedict)

test_loss = 0
with torch.no_grad():
    for feats, labels in test_dataloader:
        count = len(labels)
        loss = criterion(model(feats), labels)
        test_loss = loss.cpu().item() * len(labels)
    test_loss = test_loss/len(test_dataloader.dataset)
print("Average MSE:", test_loss)
```

```
0.5740793323516846
```

Our model can predict the test set well with an average of 0.57 error in the ratings.

In order to predict the rating for a particular course, we need to have both the employee ID and course ID to be fed into the model. We will demonstrate this for employee Harriot Laflin by predicting on all courses that he has not attended. After collecting all the rating predictions, we can use argsort to get the indexes and sort them by descending value. This will allow us to extract the highest rated five courses for him.

```
emp_to_predict="Harriot Laflin"

#Get employee ID for the employee name
pred_emp_id=emp_list[emp_list['EmpName'] == emp_to_
↪predict]["EmployeeID"].iloc[0]

#find Courses already taken by employee. We dont want␣
↪to predict those.
completed_courses=ratings_data[ratings_data[
↪"EmployeeID"] == pred_emp_id]["CourseID"].unique()

#Courses not taken by employee
new_courses = course_list.query("CourseID not in␣
↪@completed_courses")["CourseID"]

#Create a list with the same employee ID repeated for␣
↪the same number of times as the
#number of new courses. This provides the employee␣
↪and course Series with same size
emp_dummy_list=pd.Series(np.array([pred_emp_id for i␣
↪in range(len(new_courses))]))

#Predict ratings for the new courses for this employee
query_courses = torch.tensor(np.stack((emp_dummy_list.
↪to_numpy(), new_courses.to_numpy())).transpose())
projected_ratings = model(query_courses).detach().
↪cpu().numpy()
flat_ratings = projected_ratings.flatten()

print("Course Ratings: ", flat_ratings)

#Recommend top 5 courses
print("\nRating  CourseID CourseName\n----------------
↪--------------------")
for idx in (-flat_ratings).argsort()[:5]:
    course_id=new_courses.iloc[idx]
    course_name=course_list.query("CourseID ==␣
↪@course_id")["CourseName"].iloc[0]
    print(" ", round(flat_ratings[idx],1),"    ",␣
↪course_id, "   ", course_name)
```

```
Course Ratings:  [3.1841657 2.7555847 3.086042  3.3539135 3.1158957 3.
↪5255125 3.583489
 3.2019932 3.8597908 3.9257545 3.8376594 2.8970377 2.7350543 2.8152738
```

(continues on next page)

(continued from previous page)

```
 3.6061487 3.576098  2.9578328 2.6336784 3.0504766 3.4058409 2.6937842
 3.0530422 3.5699735]

Rating  CourseID CourseName
-----------------------------------
  3.9      11     Analytical Reasoning
  3.9      10     Mobile Development
  3.8      12     People Management
  3.6      16     Audio Production
  3.6      8     IT Architecture
```

As you can see, our model recommends the courses Analytical Reasoning, Mobile Development, People Management, Audio Production, and IT Architecture for Harriot Laflin.

### *9.3.4 Employee Attrition Prediction*

Because recruitment is a costly process and top talents are in short supply, it is critical that onboarded employees stay with the firm for as long as feasible. HR is responsible for employee training and development, whether for a new recruit or an experienced employee, in cooperation with management. Given the increasingly interconnected nature of our society, it is now a lot simpler for competitors to snatch away top employees. In order to retain the company's talents, human resource departments create a variety of initiatives to encourage employee loyalty and retention. Retaining employees requires significant effort to facilitate career growth, provide mentorship, perform succession planning, and facilitate interdepartmental transfers. Through the use of AI, we can use data collected from internal surveys to predict the likelihood that an employee would leave the organization. This can be useful in identifying individuals who are at risk of leaving and preventive measures can be taken to alleviate the problems that they may face.

In this example, we will use the dataset from the link below [4]:

https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

This dataset contains various survey results and personal information collected from various employees within the company. Employees who have left will be marked with "Yes" under the attrition column. We will use this historical information to train a model to be able to predict how likely will a particular employee leave the company.

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
```

(continues on next page)

(continued from previous page)

```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split,␣
↪StratifiedKFold
from sklearn.metrics import accuracy_score, confusion_
↪matrix, balanced_accuracy_score
import matplotlib.pyplot as plt
```

Let us take a look at the data.

```
employee_attrition_data = pd.read_csv("Human_
↪Resources/WA_Fn-UseC_-HR-Employee-Attrition.csv")
employee_attrition_data.head()
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumber  \
0                 1          2  Life Sciences              1               1
1                 8          1  Life Sciences              1               2
2                 2          2          Other              1               4
3                 3          4  Life Sciences              1               5
4                 2          1        Medical              1               7

   ...  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  ...                         1            80                 0
1  ...                         4            80                 1
2  ...                         2            80                 0
3  ...                         3            80                 0
4  ...                         4            80                 1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance  YearsAtCompany  \
0                  8                      0               1               6
1                 10                      3               3              10
2                  7                      3               3               0
3                  8                      3               3               8
4                  6                      3               3               2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                        0                     5
1                  7                        1                     7
2                  0                        0                     0
3                  7                        3                     0
4                  2                        2                     2

[5 rows x 35 columns]
```

```
employee_attrition_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

We have a total of 34 features collected and 1 “Attrition” ground truth column. Here is the explanation for what each number represents in the categorical features collected.

| Variable | Key |
|---|---|
| Education | 1 = 'Below College', 2 = 'College', 3 = 'Bachelor', 4 = 'Master', 5 = 'Doctor' |
| Environment Satisfaction | 1 = 'Low', 2 = 'Medium', 3 = 'High', 4 = 'Very High' |
| Job Involvement | 1 = 'Low', 2 = 'Medium', 3 = 'High', 4 = 'Very High' |
| Job Satisfaction | 1 = 'Low', 2 = 'Medium', 3 = 'High', 4 = 'Very High' |
| Performance Rating | 1 = 'Low', 2 = 'Good', 3 = 'Excellent', 4 = 'Outstanding' |
| Relationship Satisfaction | 1 = 'Low', 2 = 'Medium', 3 = 'High', 4 = 'Very High' |
| Work Life Balance | 1 = 'Bad', 2 = 'Good', 3 = 'Better', 4 = 'Best' |

Let us begin by mapping the binary variables into numbers.

```
employee_attrition_data["IsOver18"] = employee_
↪attrition_data["Over18"].map({"Y": 1, "N":0})
employee_attrition_data["DidOverTime"] = employee_
↪attrition_data["OverTime"].map({"Yes": 1, "No":0})
employee_attrition_data["IsMale"] = employee_
↪attrition_data["Gender"].map({"Male":1, "Female":0})
employee_attrition_data["Quit"] = employee_attrition_
↪data["Attrition"].map({"Yes": 1, "No": 0})

df = employee_attrition_data.drop(["Over18", "OverTime
↪", "Gender", "Attrition"], axis=1)
```

For features with multiple categories such as Job Role, we will need to create dummy variables. Dummy variables are set to 1 if the employee belongs to a specific category, such as being a Healthcare Representative, and 0 if they do not.

```
df = pd.get_dummies(df)
df.head()
```

```
   Age  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  \
0   41       1102                 1          2              1               1
1   49        279                 8          1              1               2
2   37       1373                 2          2              1               4
3   33       1392                 3          4              1               5
4   27        591                 2          1              1               7

   EnvironmentSatisfaction  HourlyRate  JobInvolvement  JobLevel  ...  \
0                        2          94               3         2  ...
1                        3          61               2         2  ...
2                        4          92               2         1  ...
3                        4          56               3         1  ...
4                        1          40               3         1  ...

   JobRole_Laboratory Technician  JobRole_Manager  \
0                              0                0
```

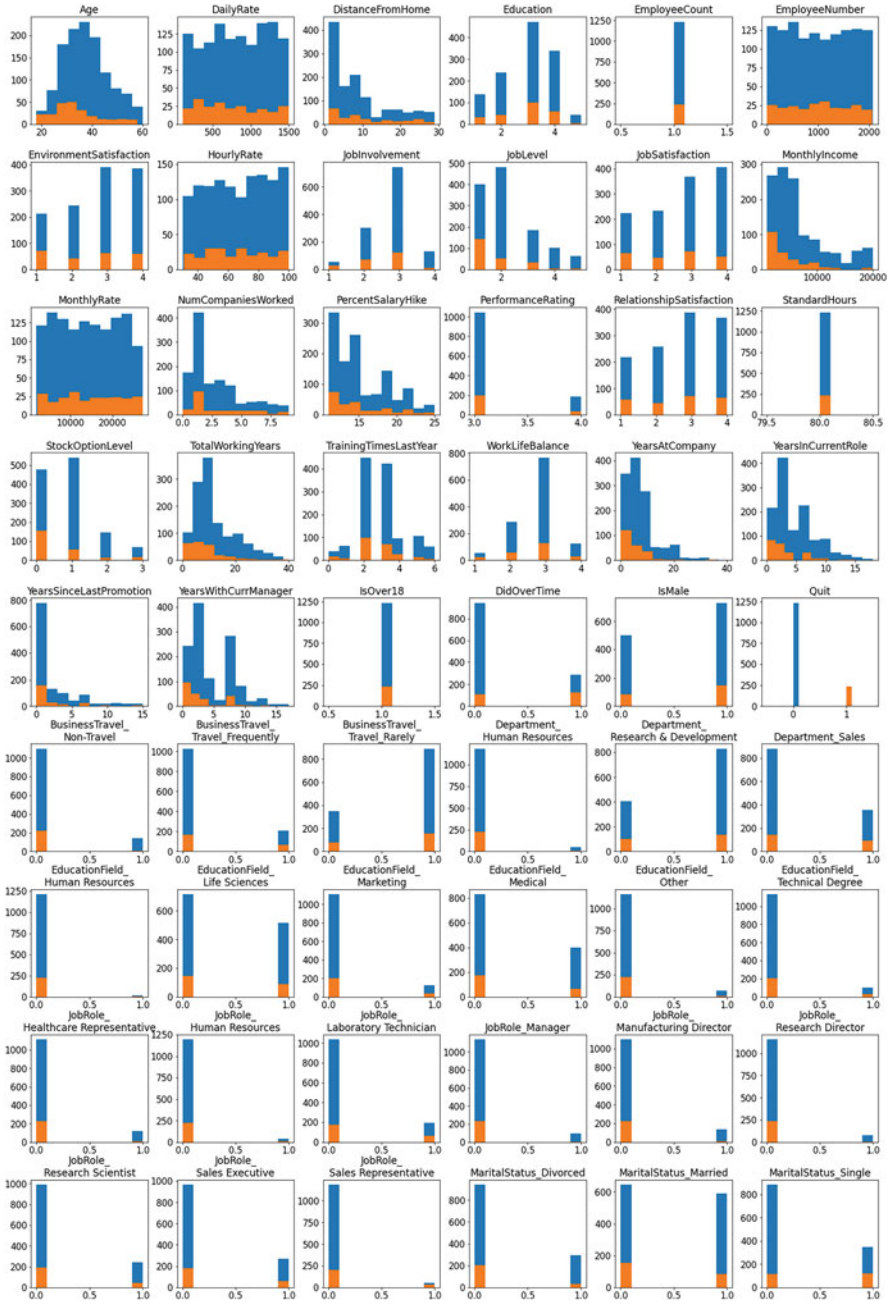
(continues on next page)

(continued from previous page)

```
1                                0                       0
2                                1                       0
3                                0                       0
4                                1                       0

   JobRole_Manufacturing Director  JobRole_Research Director  \
0                                0                          0
1                                0                          0
2                                0                          0
3                                0                          0
4                                0                          0

   JobRole_Research Scientist  JobRole_Sales Executive  \
0                           0                        1
1                           1                        0
2                           0                        0
3                           1                        0
4                           0                        0

   JobRole_Sales Representative  MaritalStatus_Divorced  \
0                             0                       0
1                             0                       0
2                             0                       0
3                             0                       0
4                             0                       0

   MaritalStatus_Married  MaritalStatus_Single
0                      0                     1
1                      1                     0
2                      0                     1
3                      1                     0
4                      1                     0

[5 rows x 54 columns]
```

Now, let us plot our features, and we will overlay the features of those that quit over those that did not. This will allow us to explore and see if any feature is highly correlated with attrition.

```
fig, axes = plt.subplots(9, 6, figsize=(20,30))
df[df["Quit"] == 0].hist(figsize=(15,15), ax=axes,␣
↪grid=False)
df[df["Quit"] == 1].hist(figsize=(15,15), ax=axes,␣
↪grid=False)
plt.tight_layout()
plt.show()
```


Age
DailyRate
DistanceFromHome
Education
EmployeeCount
EmployeeNumber
EnvironmentSatisfaction
HourlyRate
JobInvolvement
JobLevel
JobSatisfaction
MonthlyIncome
MonthlyRate
NumCompaniesWorked
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
StandardHours
StockOptionLevel
TotalWorkingYears
TrainingTimesLastYear
WorkLifeBalance
YearsAtCompany
YearsInCurrentRole
YearsSinceLastPromotion
YearsWithCurrManager
IsOver18
DidOverTime
IsMale
Quit
BusinessTravel_ Non-Travel
BusinessTravel_ Travel_Frequently
BusinessTravel_ Travel_Rarely
Department_ Human Resources
Department_ Research & Development
Department_Sales
EducationField_ Human Resources
EducationField_ Life Sciences
EducationField_ Marketing
EducationField_ Medical
EducationField_ Other
EducationField_ Technical Degree
JobRole_ Healthcare Representative
JobRole_ Human Resources
JobRole_ Laboratory Technician
JobRole_Manager
JobRole_ Manufacturing Director
JobRole_ Research Director
JobRole_ Research Scientist
JobRole_ Sales Executive
JobRole_ Sales Representative
MaritalStatus_Divorced
MaritalStatus_Married
MaritalStatus_Single

From this plot, we can learn a lot about our data. For example, the proportion of sales representatives that remain is only slightly higher than those that have quit their jobs. Thus, the sales representative role in the company has a significant attrition issue. Furthermore, we can infer that younger staff and lower paid staff are more likely to quit. Let us take a look at the how likely is a sales representative to quit.

```
ratio = employee_attrition_data.groupby("JobRole")[
↪"Quit"].sum() / employee_attrition_data.groupby(
↪"JobRole")["Quit"].count()
ratio
```

```
JobRole
Healthcare Representative    0.068702
Human Resources              0.230769
Laboratory Technician        0.239382
Manager                      0.049020
Manufacturing Director       0.068966
Research Director            0.025000
Research Scientist           0.160959
Sales Executive              0.174847
Sales Representative         0.397590
Name: Quit, dtype: float64
```

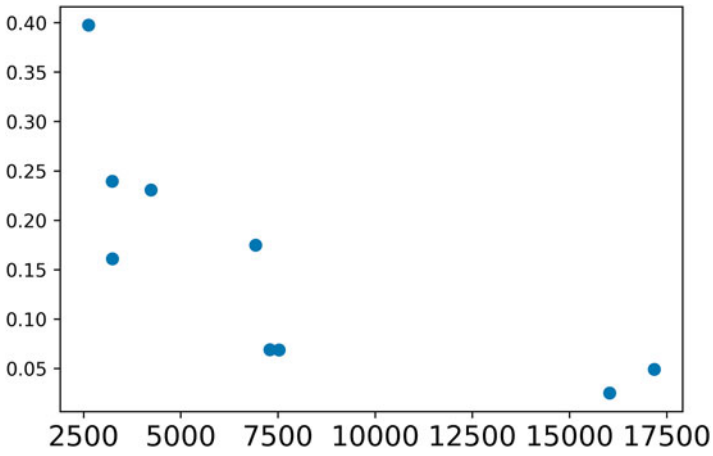
Close to 40% of sales representatives eventually quit. That is very high. Let us take a look at the salary by jobs.

```
mean_salary = employee_attrition_data.groupby("JobRole
↪")["MonthlyIncome"].mean()
mean_salary
```

```
JobRole
Healthcare Representative     7528.763359
Human Resources               4235.750000
Laboratory Technician         3237.169884
Manager                      17181.676471
Manufacturing Director        7295.137931
Research Director            16033.550000
Research Scientist            3239.972603
Sales Executive               6924.279141
Sales Representative          2626.000000
Name: MonthlyIncome, dtype: float64
```

Sales representatives have a low paying job overall which may contribute to the attrition issue. This is better visualized through a scatter plot of attrition against salary.

```
plt.scatter(mean_salary, ratio)
plt.show()
```




Now, let us split our dataset and train our model.

```
features = df.drop("Quit", axis=1)
labels = df["Quit"]
X_train, X_test, y_train, y_test = train_test_
↪split(features, labels, test_size=0.20, random_
↪state=42)

model = XGBClassifier(random_state = 0)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
0.8809523809523809
[[247   8]
 [ 27  12]]
```

We have achieved an accuracy of 88% on our dataset.

```
balanced_accuracy_score(y_test, model.predict(X_test))
```

```
0.6381598793363499
```

This can be further improved by using SMOTE to synthesize more examples of the minority class. SMOTE does this by interpolating between the features of the minority class.

```
from imblearn.over_sampling import SMOTE
```

```
oversample = SMOTE(random_state=42)
X_train, y_train = oversample.fit_resample(X_train, y_
↪train)

model = XGBClassifier(random_state = 0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, model.predict(X_test)))
print(balanced_accuracy_score(y_test, y_pred))
```

```
0.8775510204081632
[[243  12]
 [ 24  15]]
0.6687782805429864
```

It looks like our model's accuracy dropped slightly, but we are better able to predict which staff are likely to quit. Let us perform k-fold cross validation to select the best split to further improve our performance.

```
import numpy as np
from sklearn.model_selection import KFold
from copy import deepcopy
kf = KFold(n_splits=5, shuffle=True, random_state=42)

selected_model = None
best_score = -1
best_split = None
for train, test in kf.split(df):
    X_train = df.iloc[train].drop("Quit", axis=1)
    y_train = df.iloc[train]["Quit"]
    oversample = SMOTE(random_state=0)
    X_train, y_train = oversample.fit_resample(X_
↪train, y_train)
```

(continues on next page)

(continued from previous page)

```
    model = XGBClassifier(use_label_encoder=False,␣
↪verbosity=0)
    model.fit(X_train, y_train)
    X_test = df.iloc[test].drop("Quit", axis=1)
    y_test = df.iloc[test]["Quit"]

    score = balanced_accuracy_score(y_test, model.
↪predict(X_test))
    if score > best_score:
        best_score = score
        selected_model = deepcopy(model)
        best_split = X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = best_split
print(accuracy_score(y_test, selected_model.predict(X_
↪test)))
print(confusion_matrix(y_test, selected_model.
↪predict(X_test)))
print(balanced_accuracy_score(y_test, selected_model.
↪predict(X_test)))
```

```
0.8741496598639455
[[236   7]
 [ 30  21]]
0.6914790607601065
```

Now we have a model that performs better in predicting attrition with a balanced accuracy score of 69% while maintaining a high accuracy score of 87.4%.

## Exercises

1. List at least three different roles and responsibilities of human resource departments.
2. List three different challenges that human resource departments face.
3. For each of the challenges listed in 2, identify one way that artificial intelligence could potentially be used to alleviate the problem.
4. Predict employee attrition for healthcare workers.
   We will utilize this dataset [7]:
   https://www.kaggle.com/datasets/jpmiller/employee-attrition-for-healthcare
   This dataset consists of information collected about various employees working in the healthcare sector and a label for whether there was attrition or not.

In this exercise, try to use what you have learned so far to develop a model that can predict whether an employee is likely to leave the company.

## References


1. Aryashah2k (2022) Hr analytics. In: Kaggle. https://www.kaggle.com/datasets/aryashah2k/datasets-in-hr-analytics-applied-ai. Accessed 27 Apr 2023
2. Aryashah2k (2022) Recommending employee training courses. https://www.kaggle.com/code/aryashah2k/recommending-employee-training-courses. Accessed 27 Apr 2023
3. Christina P (2022) Ai in HR: benefits, limitations, and challenges of artificial intelligence in the workplace. https://www.efrontlearning.com/blog/2022/05/ai-in-hr.html. Accessed 27 Apr 2023
4. IBM (2017) Ibm hr analytics employee attrition & performance. In: Kaggle. https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset. Accessed 27 Apr 2023
5. Lokkagle (2020) Glassdoor data scientist salary prediction. https://www.kaggle.com/code/lokkagle/glassdoor-data-scientist-salary-prediction/notebook. Accessed 27 Apr 2023
6. Melanie J (2022) 7 effective uses of AI in recruitment. https://www.unleash.ai/artificial-intelligence/7-effective-uses-of-ai-in-recruitment/. Accessed 27 Apr 2023
7. Miller J (2022) Employee attrition for healthcare. https://www.kaggle.com/datasets/jpmiller/employee-attrition-for-healthcare
8. Rafunlearnhub (2021) Recruitment data. In: Kaggle. https://www.kaggle.com/datasets/rafunlearnhub/recruitment-data. Accessed 27 Apr 2023
9. Rafunlearnhub (2021) should recruit or not. https://www.kaggle.com/code/rafunlearnhub/should-recruit-or-not. Accessed 27 Apr 2023
10. Shweta (2022) What is human resources? The ultimate guide. https://www.forbes.com/advisor/business/what-is-human-resources/. Accessed 27 Apr 2023
11. VectorMine (2023) Human resource management employee job Research stock vector (royalty free) 2010439232 | Shutterstock. https://www.shutterstock.com/image-vector/human-resource-management-employee-job-research-2010439232. Accessed 13 May 2023

# Chapter 10
# AI in Sales


**Learning Outcomes**

- Understand the roles and responsibilities of sales departments.
- Be able to explain the various segments of the sales cycle.
- Identify ways that artificial intelligence can be applied to business sales.
- Develop artificial intelligence solutions for lead scoring, chatbots, and recommending products.

## 10.1 Introduction to Sales

Sales is the process of convincing a potential customer to purchase goods and services. The typical sales process usually requires salespeople to contact leads that fit the profile of their target audience. These leads are often identified by the marketing department first, before being handed over to the sales team. The sales team will then qualify the leads produced by the marketing team. Leads that are qualified are called Sales Qualified Leads and are ready to be approached by the sales team. The salesperson will then persuade the lead to make a purchase by highlighting a problem that would be fixed by using the product.

This is achieved by developing and executing a sales plan. Sales strategies lay out explicit guideline and targets for the sales teams to follow. These strategies are developed to help maximize sales and also ensure that sales teams are on the same page. Prospecting, lead qualification, and developing persuasive messaging to demonstrate product value to potential buyers are all essential components of a successful sales strategy.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_10

Sales representatives are often tasked to:

1. **Convert prospective customers**
   The fundamental objective of any company's sales team is to boost the company's profits by generating more revenue. A successful sales team will be able to achieve a high conversion rate and boost the company's profits.
2. **Grow the business by building relationships**
   When the vast majority of your clientele are pleased with the service they receive, news quickly spreads about your company. New customers will be easier to convert into paying customers if they can readily obtain glowing recommendations from your present customers. An effective sales team establishes the framework for future expansion by fostering constructive, long-lasting connections with clients.
3. **Retain existing customers**
   The cost of bringing in new consumers far outweighs the cost of keeping old ones. That is why it is so important for sales and account teams to follow up with customers after the transaction to make sure they are satisfied.

### *10.1.1 The Sales Cycle*

The sales cycle in Fig. 10.1 represents the various stages that a salesperson would have to accomplish in order to convert a client from a prospect into a customer. Good sales would eventually lead to more positive word of mouth advertisements and repeat purchases, thus restarting the sales cycle once more.

1. **Prospecting**
   The first stage of the sales cycle begins with prospecting leads. In this stage you will identify ideal customers and figure out a strategy to best position your product or services to them. You will also have to identify and compile a list of prospective leads to contact once the campaign has started.
   You may want to include the lead's company, position, and other information in your list and categorize them by their industry or other factors.
2. **Making Contact**
   Getting in touch with prospective clients is the next step. There are many ways to do this—via email, phone, face-to-face meetings, etc. These methods are useful but only if potential leads can see the value of your products instantly after contact. Start by establishing contact, convincing them of the value you are able to provide, and arranging a formal meeting with them.
3. **Qualifying Leads**
   This stage happens after the first meeting with a prospective lead. Not everyone you add to the leads list may be a good fit, suitable leads are otherwise known as a qualified lead. This is why you will need a strong plan for qualifying your leads. By only contacting sales qualified leads who have shown interest in purchasing your products or services, you can save a lot of time and energy.


**Fig. 10.1** The sales cycle

Ask questions to get a better understanding of the potential customer's needs and identify areas which you might be able to help with in the first meeting. Their expected schedule and budget are two important pieces of information to consider. This will help you assess whether working with them will be beneficial.

4. **Lead Nurturing**
   Once leads have been successfully qualified, the next step is to begin nurturing them. Begin by identifying which of your leads are most likely to convert and start providing them with targeted information about your products. This can be easily done through email or social media channels.
   However, lead nurturing entails more than simply providing prospective leads with brand-related information. This process will entail convincing the lead that they need a product or service similar to what your business offers.
5. **Making an Offer**
   Presenting offers to prospective leads is a critical next step in the sales cycle. This implies sales reps need to convince clients to believe your product or service is the greatest possible option for them. This stage not only requires the most amount of work but will also provide a possibility of conversion if executed well.

6. **Handling Objections**
   When presented with an offer, the vast majority of your potential consumers will either reject it outright or flood the sales representatives with inquiries. Therefore, sales teams have to respond quickly and effectively to address their questions and concerns.
   Some of the common questions that might arise are "Why is the price so high?" or "What sets you apart from the competition/what is your unique selling proposition (USP)?" You must be ready to respond confidently and calmly to their questions.
7. **Closing a Sale**
   If all the procedures up to this point were well executed, sales teams will have an easier time with this step. The outcome of this stage is dependent on how the previous stages were executed.
   The strategy you should adopt in this stage is dependent on how your clients established a connection with your brand in the previous stages. A direct approach is appropriate if you are confident that you have addressed all of their issues and that they are actively listened to what you have to say. Otherwise, if you believe they are not being adequately nurtured, a softer approach would be appropriate.
8. **Delivery**
   After closing a sale, the process does not stop. You will still need to put in the effort to continuously win over your customer. Good delivery of your product or service can lead to word-of-mouth advertising and more purchases from satisfied customers. This is your opportunity to impress them and deliver on the promises you made earlier.

## 10.2 Artificial Intelligence in Sales

The concept of highly streamlined sales processes that are powered by artificial intelligence and machine learning is not just a fantasy; it is already a reality [3].

Through leveraging robust algorithms and in-depth data, intelligent, integrated AI-enabled sales solutions have the potential to enhance decision-making, boost sales rep productivity, and boost the efficiency of sales processes to produce a superior customer experience.

Including artificial intelligence enabled tools to revamp your sales process presents a new opportunity to differentiate yourself from the competition. Currently, firms are trying to align their sales processes with manners which customers are comfortable in.

A sustainable competitive advantage can be gained by businesses that identify, disseminate, and use best-selling techniques. This means that the vendors' performance is now the single most important factor in determining victory rates.

To some extent, artificial intelligence may provide a problem for sales teams all by itself. However, AI has great promise for increased productivity, effectiveness, and sales success when integrated with a well-thought-out strategy.

Ways that AI is Changing Sales:

1. Lead Scoring
2. Price Optimization
3. Managing for Performance
4. Upselling and Cross-selling
5. Forecasting
6. Customer Improvement
7. Practice Improvement
8. Easier Prioritization
9. Increased Sales
10. Decreased Costs and Time
11. Increased Human Touch

## 10.3 Applications of AI in Sales

### *10.3.1 Lead Scoring*

Lead scoring is the process of assigning a score to each generated lead [6]. Most businesses use a point system: the better the fit and interest of the lead, the higher the lead's score. Lead scoring helps marketing and sales focus their efforts so that they are always concentrating on the most promising leads to convert.

Inbound marketing is great because it brings in leads. When your inbound marketing approach is in full swing, you will notice a steady flow of qualified leads into your website and inbox.

However, not all leads are created equal, and your sales team only has so many hours in a day to close them. Lead scoring provides your marketing and sales teams with a standardized scoring framework to determine which leads to prioritize.

Therefore, lead scoring is necessary to help sales teams avoid burnout and help them focus on closing more relevant leads.

Most lead scoring models utilize a point system, whereby points are assigned based on attributes or qualities that a lead has. Points can be either positive or negative. As points are tallied for each lead, those with the highest score will be the most qualified.

The fit of a lead refers to how well they suit your product or service. If they are in your service region, operate in the appropriate industry, and have the appropriate job title or role at their organization, then they likely match your buyer profile. They are probably an excellent fit for your product or service.

However, even if a lead is a perfect match, they are not a high-quality lead if they lack interest in your organization, product, or services. When assessing leads, it is

essential to consider both a lead's suitability and interest. Webpage visits, interacting with offers and looked at your pricing pages are some ways to quantify a buyer's interest in a product or service.

Lead scoring makes it simple to determine which incoming leads are most likely to convert. Therefore, your sales force will spend less time nurturing leads and more time closing deals.

When they are able to see at a glance the best qualified leads, they can invest their time in a way that will bring the largest impact to your bottom line. This would be accomplished by reaching out to and interacting with those leads first.

Lead scoring is useful for businesses that want to focus their efforts on prospects which will yield the most return.

Developing a lead scoring strategy that works for your firm requires some effort up front, but the result is that your marketing and sales teams are:

1. Aligned on which leads are the most valuable.
2. Concentrated on leads that are most likely to result in a profit.

In this dataset, an education company, XEducation, sells online courses. In this example we will create a lead scoring system to help recognizing and selecting promising leads that will most likely convert to paying customers. This will be done by assigning lead scores to each lead such that customers with a high lead score will have a higher chance of conversion.

The dataset can be obtained from [2]:
https://www.kaggle.com/datasets/ashydv/leads-dataset

```
# https://www.kaggle.com/code/adityamishra0708/lead-
↪scoring/notebook
import pandas as pd
import numpy as np


leads = pd.read_csv('Sales_Marketing/Leads.csv')
leads.head()
```

```
                            Prospect ID  Lead Number              Lead Origin  \
0  7927b2df-8bba-4d29-b9a2-b6e0beafe620       660737                      API
1  2a272436-5132-4136-86fa-dcc88c88f482       660728                      API
2  8cc8c611-a219-4f35-ad23-fdfd2656bd8a       660727  Landing Page Submission
3  0cc2df48-7cf4-4e39-9de9-19797f9b38cc       660719  Landing Page Submission
4  3256f628-e534-4826-9d63-4a8b88782852       660681  Landing Page Submission

      Lead Source Do Not Email Do Not Call  Converted  TotalVisits  \
0      Olark Chat           No          No          0          0.0
1  Organic Search           No          No          0          5.0
2  Direct Traffic           No          No          1          2.0
3  Direct Traffic           No          No          0          1.0
4          Google           No          No          1          2.0

   Total Time Spent on Website  Page Views Per Visit  ...  \
0                            0                   0.0  ...
1                          674                   2.5  ...
```

(continues on next page)

(continued from previous page)

```
2                         1532                  2.0  ...
3                          305                  1.0  ...
4                         1428                  1.0  ...

  Get updates on DM Content    Lead Profile    City  \
0                        No          Select  Select
1                        No          Select  Select
2                        No  Potential Lead  Mumbai
3                        No          Select  Mumbai
4                        No          Select  Mumbai

  Asymmetrique Activity Index Asymmetrique Profile Index  \
0                  02.Medium                  02.Medium
1                  02.Medium                  02.Medium
2                  02.Medium                    01.High
3                  02.Medium                    01.High
4                  02.Medium                    01.High

  Asymmetrique Activity Score Asymmetrique Profile Score  \
0                        15.0                        15.0
1                        15.0                        15.0
2                        14.0                        20.0
3                        13.0                        17.0
4                        15.0                        18.0

  I agree to pay the amount through cheque  \
0                                       No
1                                       No
2                                       No
3                                       No
4                                       No

  A free copy of Mastering The Interview Last Notable Activity
0                                     No              Modified
1                                     No         Email Opened
2                                    Yes         Email Opened
3                                     No              Modified
4                                     No              Modified

[5 rows x 37 columns]
```

Let us view the columns in our dataset

```
leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 37 columns):
 #   Column                                         Non-Null Count  Dtype
---  ------                                         --------------  -----
 0   Prospect ID                                    9240 non-null   object
 1   Lead Number                                    9240 non-null   int64
 2   Lead Origin                                    9240 non-null   object
 3   Lead Source                                    9204 non-null   object
 4   Do Not Email                                   9240 non-null   object
 5   Do Not Call                                    9240 non-null   object
 6   Converted                                      9240 non-null   int64
 7   TotalVisits                                    9103 non-null   float64
 8   Total Time Spent on Website                    9240 non-null   int64
 9   Page Views Per Visit                           9103 non-null   float64
 10  Last Activity                                  9137 non-null   object
 11  Country                                        6779 non-null   object
```

(continues on next page)

(continued from previous page)

```
 12  Specialization                                 7802 non-null   object
 13  How did you hear about X Education             7033 non-null   object
 14  What is your current occupation                6550 non-null   object
 15  What matters most to you in choosing a course  6531 non-null   object
 16  Search                                         9240 non-null   object
 17  Magazine                                       9240 non-null   object
 18  Newspaper Article                              9240 non-null   object
 19  X Education Forums                             9240 non-null   object
 20  Newspaper                                      9240 non-null   object
 21  Digital Advertisement                          9240 non-null   object
 22  Through Recommendations                        9240 non-null   object
 23  Receive More Updates About Our Courses         9240 non-null   object
 24  Tags                                           5887 non-null   object
 25  Lead Quality                                   4473 non-null   object
 26  Update me on Supply Chain Content              9240 non-null   object
 27  Get updates on DM Content                      9240 non-null   object
 28  Lead Profile                                   6531 non-null   object
 29  City                                           7820 non-null   object
 30  Asymmetrique Activity Index                    5022 non-null   object
 31  Asymmetrique Profile Index                     5022 non-null   object
 32  Asymmetrique Activity Score                    5022 non-null   float64
 33  Asymmetrique Profile Score                     5022 non-null   float64
 34  I agree to pay the amount through cheque       9240 non-null   object
 35  A free copy of Mastering The Interview         9240 non-null   object
 36  Last Notable Activity                          9240 non-null   object
dtypes: float64(4), int64(3), object(30)
memory usage: 2.6+ MB
```

Now we will need to drop:

1. Irrelevant columns as they do not provide useful information.
2. Columns with too many incomplete information for simplicity instead of figuring out how to fill missing values.
3. Drop columns with 99% of all answers being a single option as this does not help discriminate between leads.

```
#drop prospect id, lead number, country and city as␣
↪it is not helpful in lead scoring
leads.drop(['Prospect ID', 'Lead Number', 'Country',␣
↪'City'], axis = 1, inplace = True)

#dropping columnns
cols = leads.columns

for i in cols:
    # Removing the columns that contain more than 30%␣
↪of missing values
    if((100*leads[i].isnull().sum()/len(leads.index))␣
↪>= 30):
        leads.drop(i, axis = 1, inplace = True)
        continue
    # Removing columns which 99% of all values are 1␣
↪option
```

(continues on next page)

(continued from previous page)

```
    for value in leads[i].unique():
        if len(leads[leads[i] == value])/len(leads[i].
↪dropna()) > 0.99:
            leads.drop(i, axis = 1, inplace = True)
            break
```

Afterward, we are left with 11 columns only.

```
leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 14 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Lead Origin                             9240 non-null   object
 1   Lead Source                             9204 non-null   object
 2   Do Not Email                            9240 non-null   object
 3   Converted                               9240 non-null   int64
 4   TotalVisits                             9103 non-null   float64
 5   Total Time Spent on Website             9240 non-null   int64
 6   Page Views Per Visit                    9103 non-null   float64
 7   Last Activity                           9137 non-null   object
 8   Specialization                          7802 non-null   object
 9   How did you hear about X Education      7033 non-null   object
 10  What is your current occupation         6550 non-null   object
 11  Lead Profile                            6531 non-null   object
 12  A free copy of Mastering The Interview  9240 non-null   object
 13  Last Notable Activity                   9240 non-null   object
dtypes: float64(2), int64(2), object(10)
memory usage: 1010.8+ KB
```

```
leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 14 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Lead Origin                             9240 non-null   object
 1   Lead Source                             9204 non-null   object
 2   Do Not Email                            9240 non-null   object
 3   Converted                               9240 non-null   int64
 4   TotalVisits                             9103 non-null   float64
 5   Total Time Spent on Website             9240 non-null   int64
 6   Page Views Per Visit                    9103 non-null   float64
 7   Last Activity                           9137 non-null   object
 8   Specialization                          7802 non-null   object
 9   How did you hear about X Education      7033 non-null   object
 10  What is your current occupation         6550 non-null   object
 11  Lead Profile                            6531 non-null   object
 12  A free copy of Mastering The Interview  9240 non-null   object
 13  Last Notable Activity                   9240 non-null   object
```

(continues on next page)

(continued from previous page)

```
dtypes: float64(2), int64(2), object(10)
memory usage: 1010.8+ KB
```

For each of the remaining columns, let us drop the null rows.

```
cols = leads.columns

for i in cols:
    leads = leads[~pd.isnull(leads[i])]

print(len(leads))
```

```
6372
```

We now have 6000+ rows to still work with. Let us replace all the values that contain "Select" with "others."

```
leads = leads.replace('Select', 'Others')
```

We need to create one-hot categorical variables for columns which are not numerical.

```
dummy_cols= leads.loc[:, leads.dtypes == 'object']

print(dummy_cols.columns)

dummy = pd.get_dummies(leads[dummy_cols.columns],␣
↪drop_first=True)

# Add the results to the master dataframe
leads = pd.concat([leads, dummy], axis=1)

leads = leads.drop(dummy_cols.columns, 1)
```

```
Index(['Lead Origin', 'Lead Source', 'Do Not Email',
↪'Last Activity',
       'Specialization', 'How did you hear about X␣
↪Education',
       'What is your current occupation', 'Lead␣
↪Profile',
       'A free copy of Mastering The Interview',
↪'Last Notable Activity'],
      dtype='object')
```

Now we will scale numerical features to lie between 0 and 1.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
leads[['TotalVisits', 'Page Views Per Visit', 'Total␣
↪Time Spent on Website']] = scaler.fit_
↪transform(leads[['TotalVisits', 'Page Views Per␣
↪Visit', 'Total Time Spent on Website']])
leads.head()
```

```
   Converted  TotalVisits  Total Time Spent on Website  Page Views Per Visit  \
0          0     0.000000                     0.000000              0.000000
1          0     0.019920                     0.296655              0.045455
2          1     0.007968                     0.674296              0.036364
3          0     0.003984                     0.134243              0.018182
4          1     0.007968                     0.628521              0.018182

   Lead Origin_Landing Page Submission  Lead Origin_Lead Add Form  \
0                                    0                          0
1                                    0                          0
2                                    1                          0
3                                    1                          0
4                                    1                          0

   Lead Origin_Lead Import  Lead Source_Direct Traffic  Lead Source_Facebook  \
0                        0                           0                     0
1                        0                           0                     0
2                        0                           1                     0
3                        0                           1                     0
4                        0                           0                     0

   Lead Source_Google  ...  Last Notable Activity_Email Opened  \
0                   0  ...                                  0
1                   0  ...                                  1
2                   0  ...                                  1
3                   0  ...                                  0
4                   1  ...                                  0

   Last Notable Activity_Email Received  \
0                                     0
1                                     0
2                                     0
3                                     0
4                                     0

   Last Notable Activity_Had a Phone Conversation  \
0                                               0
1                                               0
2                                               0
3                                               0
4                                               0

   Last Notable Activity_Modified  \
0                               1
1                               0
2                               0
3                               1
4                               1

   Last Notable Activity_Olark Chat Conversation  \
0                                              0
1                                              0
```

(continues on next page)

(continued from previous page)

```
2                                                 0
3                                                 0
4                                                 0

   Last Notable Activity_Page Visited on Website  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   Last Notable Activity_SMS Sent  Last Notable Activity_Unreachable  \
0                               0                                   0
1                               0                                   0
2                               0                                   0
3                               0                                   0
4                               0                                   0

   Last Notable Activity_Unsubscribed  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0

   Last Notable Activity_View in browser link Clicked
0                                                   0
1                                                   0
2                                                   0
3                                                   0
4                                                   0

[5 rows x 89 columns]
```

Now that our data is all cleaned up, we can split the dataset into training and testing data. We will drop our target variable from the features, "Converted," which represents leads that were successfully converted. We will use it for our labels instead.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
↪classification_report

X = leads.drop(['Converted'], 1)
y = leads['Converted']

# Split the dataset into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X,
↪ y, train_size=0.7, test_size=0.3, random_state=100)
```

Now let us train our model.

```
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X_train,y_train)

print("Training Accuracy")
print(logreg.score(X_train,y_train))
print("Testing Accuracy")
print(logreg.score(X_test,y_test))

predicted = logreg.predict(X_test)
print(confusion_matrix(y_test,predicted))
print(classification_report(y_test,predicted))
```

```
Training Accuracy
0.8141255605381166
Testing Accuracy
0.8043933054393305
[[824 164]
 [210 714]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.82       988
           1       0.81      0.77      0.79       924

    accuracy                           0.80      1912
   macro avg       0.81      0.80      0.80      1912
weighted avg       0.80      0.80      0.80      1912
```

Our model has achieved a testing accuracy of 80.4%, which means we can correctly classify whether a lead will be converted or not with 80% accuracy. We can further rank the leads based on the probability scores provided by the logistic regression function where index 1 represents the probability of a lead converting. This score is our lead score. Let us view which are the top 10 leads in our test set.

```
lead_score = logreg.predict_proba(X_test)[:,1]
sorted_leads = lead_score.argsort()
# reverse the order since argsort returns results in␣
↪ascending order
sorted_leads = sorted_leads[::-1]

print(sorted_leads[0:10])
print(lead_score[sorted_leads][0:10])
```

```
[ 399  430 1323  905 1546  160   79   91  131 1744]
[0.99916925 0.99838004 0.99774375 0.99766929 0.
↪99766929 0.99763192
 0.99763192 0.99763192 0.99763192 0.99750443]
```

Great, now we know which leads our sales reps should focus on first.

### *10.3.2 Sales Assistant Chatbot*

Salespeople are under consistent pressure to generate new business. However, they are often bogged down by administrative tasks that prevent them from focusing on what really matters: closing deals and developing relationships with prospects [4].

Shortening the sales cycle will mean increasing the productivity of the sales teams, thus bringing in more sales to the company. However, improving the sales cycle is easier said than done. In order to do so, we need to be fully aware of the things happening at every stage of the cycle to identify practices that work and things that need improvement.

For example, improving the quality of your leads or identifying leads that are more likely to become loyal customers will help you save time and effort in nurturing unqualified leads.

Artificial intelligence can be utilized in sales to boost the productivity of the sales operations. This can come in the form of chatbots and virtual assistants that can help reduce the administrative and repetitive work that sales teams encounter. For example, chatbots can help ask and answer simple and repetitive questions that can reduce time spent by the sales representative.

Simple questions such as "Are you an existing customer?" ensure that chats are correctly routed to customer support instead of sales. Other questions such as "What are you looking for?" can help determine which products the customer might be interested in. With existing information about the customer, personalized chats can be tailored to improve customer experience and engagement. All of this information would allow the chatbot to route the appropriate sales team based on interest, location, industry, and many other factors.

Benefits will include:

1. **Better customer engagement and reduced churn with instant responses**
   Research indicates that leads are more likely to convert into customers if you respond quickly under the first five minutes [5]. Chatbots enable 24/7 instant conversations, and thus this allows people to engage with businesses anytime, anywhere. This allows businesses to engage with their potential customers and increase the chances of converting them into paying customers.
   Conversational commerce is also picking up via messenger apps which allows chatbots to help drive businesses and sales through platforms such as WhatsApp, Telegram, and Facebook Messenger.
2. **Customer-centric conversations**
   Personalization is a key feature of chatbots as they can analyze the customers, data, conversations, and behavior to effectively respond, recommend, or nudge them into making a purchase. This helps drive customer satisfaction and increases sales efficiency.

   For example, chatbots are able to re-target prospective customers who stop halfway while making a purchase, thus creating new opportunities with no human intervention [1]. The bot re-engages with these clients to answer their questions and convince them to make the purchase by offering exclusive deals

and discounts for a limited time. This has the potential to increase customer engagement by 15% and speed up sales by 15–20%.

3. **Allowing sales teams to better focus on selling**
   Sales teams are able to better focus on selling tasks rather than administrative duties that can be offloaded by the chatbot. This can help increase conversions in a business as sales teams are able to provide higher quality interactions with potential customers and reach out to more leads. Furthermore, conversation guidance can help recommend ideas and messages to sales representatives to help tailor to the customer and improve their sales pitches.
   After sales, customer service can also be assisted by chatbots by sending follow-up messages and collecting feedback. This will help sales teams to help improve customer satisfaction.

Here we will demonstrate how to develop a simple telegram chatbot using the SnatchBot platform.

To begin, please visit https://snatchbot.me/ and register for a free account.

After you are done, login to your account and you should be directed to your dashboard with a navigation sidebar on the left. Click on "My Bots" on the sidebar as shown in Fig. 10.2 to manage your chatbots.

This page will allow you to see all your bots.

Click on "Create Bot" as shown in Fig. 10.3 to create your first bot.

Now click on "Blank Bot" as shown in Fig. 10.4.

Decide on a name for your bot, write a description, select your language of choice, and upload a profile photo for your bot as shown in Fig. 10.5.

Now that the bot has been created, we will need to develop the chat interface.

Click on "Add new interaction/Plugin" as shown in Fig. 10.6 to create the opening message.

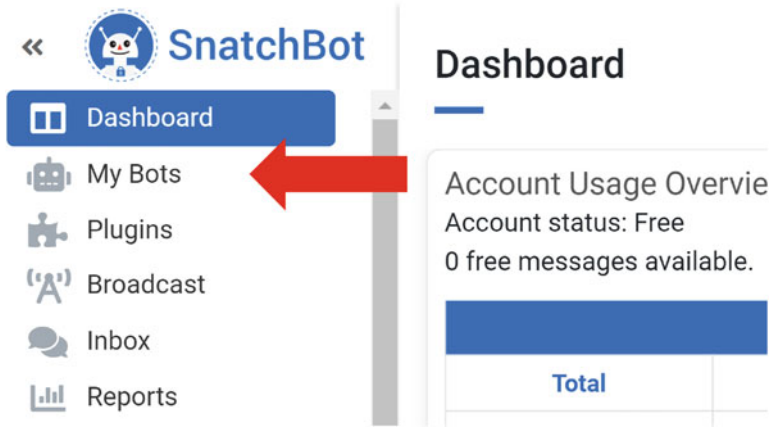



**Fig. 10.2** Click on my bots

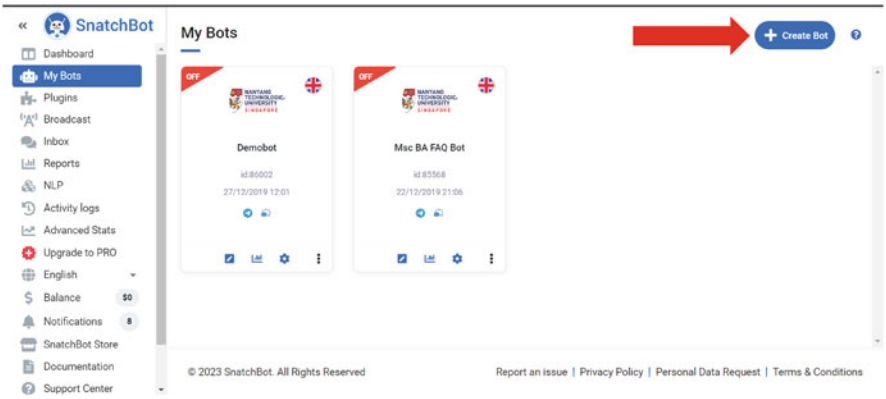



**Fig. 10.3** Click on create bot

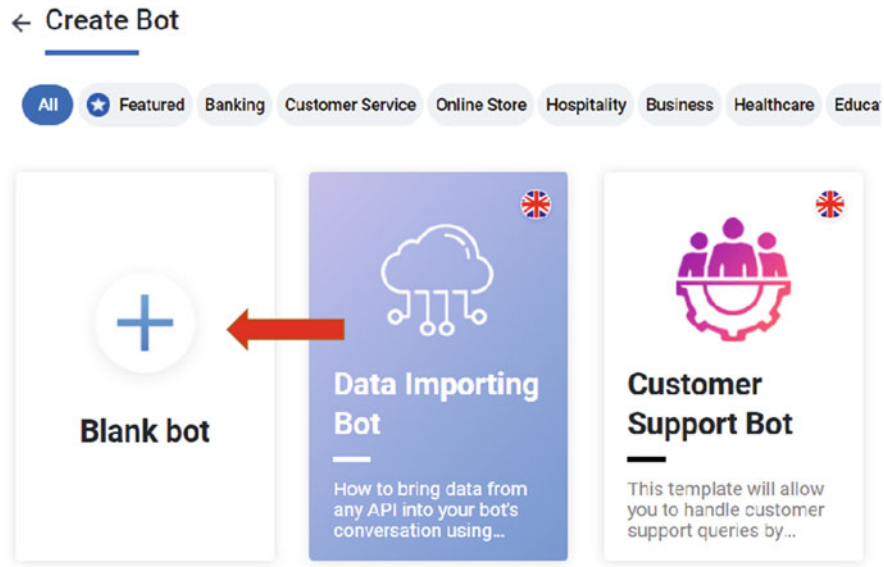



**Fig. 10.4** Click on blank bot

On this page, select the "Bot Statement" option, name the interaction "Welcome," and click "Add" as shown in Fig. 10.7.

Type in a message in the message box, shown in Fig. 10.8, you would like the customer to see. For example, you could craft a message saying "Hello welcome to XYZ store, we are a business that sells sneakers. How may I help you?"

In this example, let us assume the customer wants to know about the latest sneakers you have in stock. To do that, you will need to craft a response and create a connection.

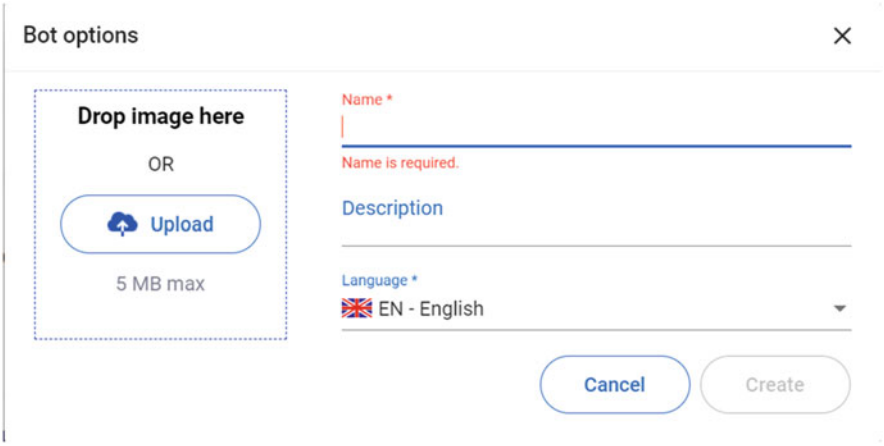



**Fig. 10.5** Fill in name

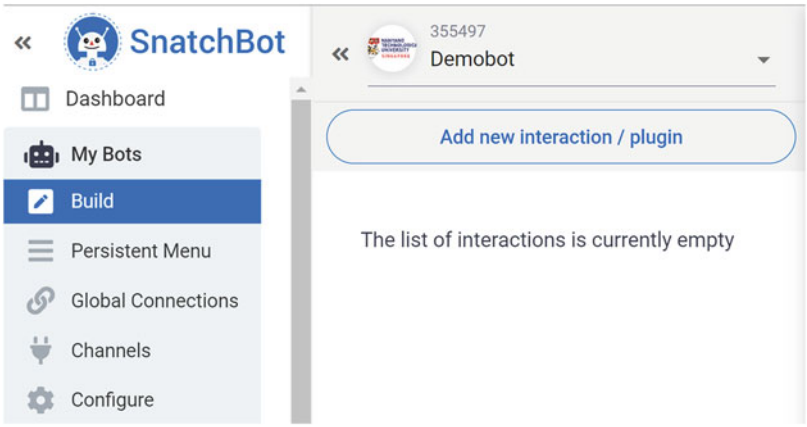



**Fig. 10.6** Add interaction

Let us create a new response by clicking on "Add new interaction / plugin" again to create another response, just like how the Welcome statement was created.

Once again, select the "Bot Statement" option, name the interaction "Response," and click "Add."

Write a response such as "Our latest sneaker additions in March are the Adidas NMD 360 shoes and the Nike Air Max 270."

Now that you have crafted a response, you will need to set up a connection.

Go back to "Welcome" (Fig. 10.9).

In the tabs shown in Fig. 10.10, select "Connections."

Type in a quick reply option such as "See New Arrivals" and click on the "+" button to add the quick reply. A prompt will ask you whether you would like to add a new connection to your quick reply. Click on "Yes" to proceed. You will see the dialog below in Fig. 10.11.

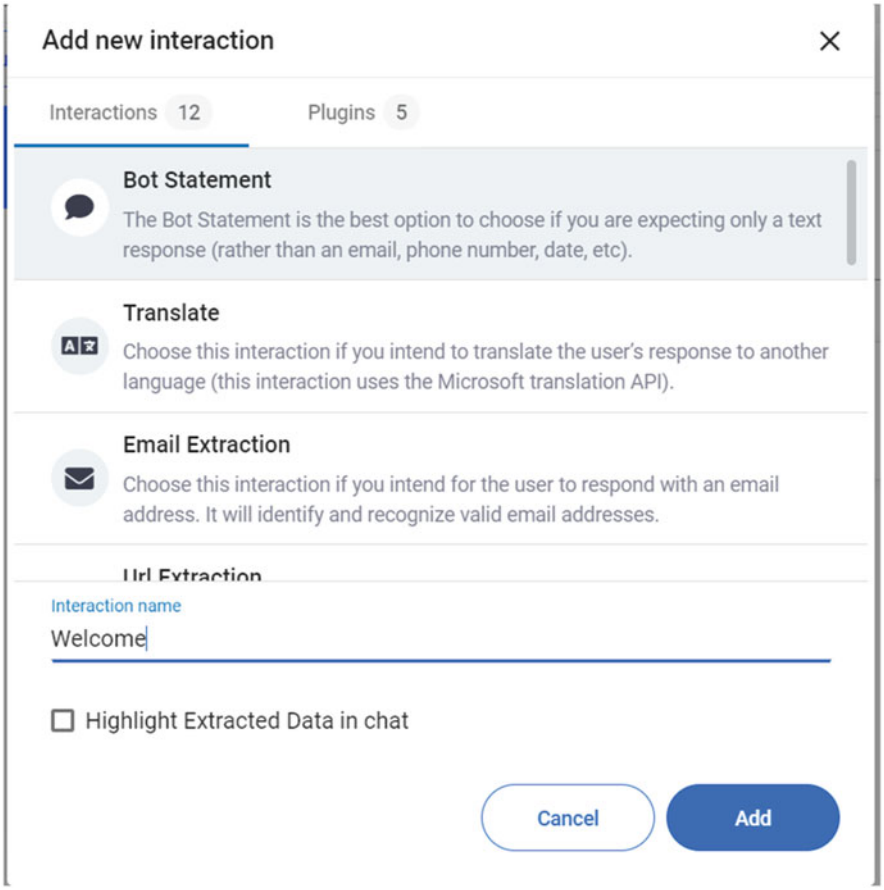



**Fig. 10.7** Add bot statement

Under Item, select "Response to this interaction." Under condition, select "contains (any part)." Under interaction, select "Response." Click "Add Quick Reply" as shown in Fig. 10.11.

Congrats! you have created your first chatbot!

Now let us deploy it on telegram.

Download telegram.

Create an account.

Go to telegram web at https://web.telegram.org/

Type BotFather in the search bar shown in Fig. 10.12.

Click on the start button, and you now will see the prompts in Fig. 10.13.

Type /new bot.

Type the name of your telegram bot, e.g., "XYZ store."

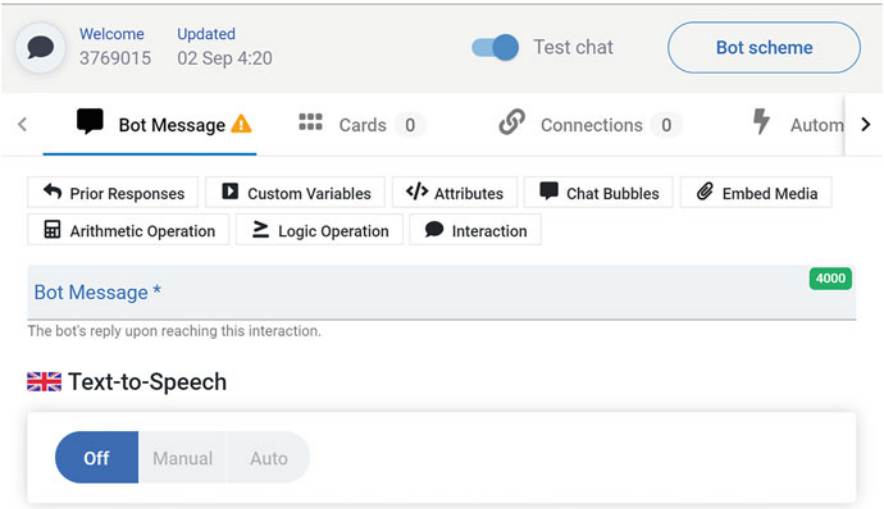



**Fig. 10.8** Add message

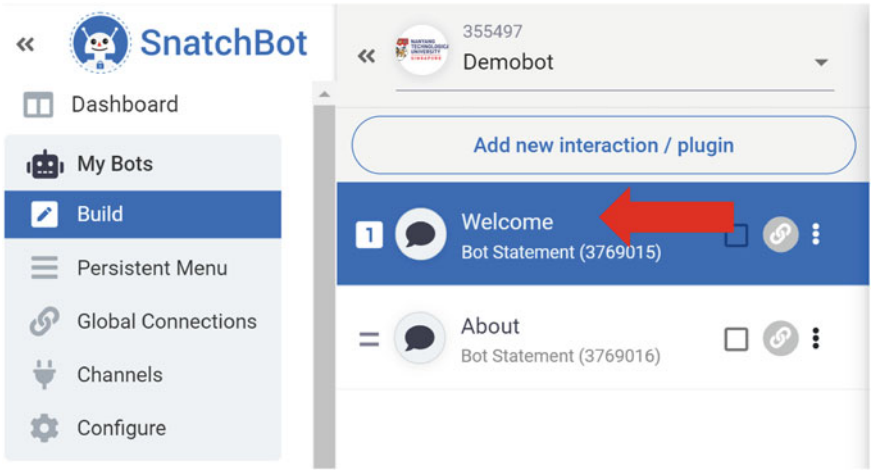



**Fig. 10.9** Click on welcome

Type the unique telegram username for your bot, and this will be how people can find your bot publicly on telegram.

Copy the token shown in Fig. 10.14.

Go back to the SnatchBot webpage.

Click on channels.

Click on telegram as shown in Fig. 10.15.

Scroll down to click on step 3.

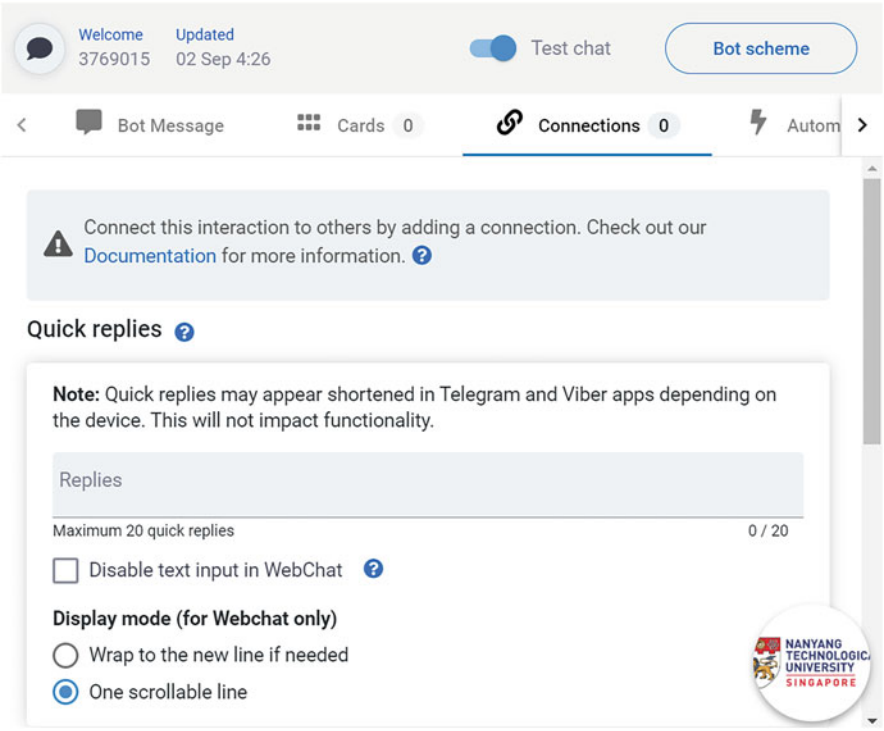



**Fig. 10.10** Select connections tab

Paste the Telegram API token into the text box as shown in Fig. 10.16, and click "Deploy."

Now your bot is running live on telegram!

In this example, we have shown how a simple chatbot can be created via the SnatchBot platform. There are many more things that can be done with chatbots.

1. Complex conversations and better insights can be enabled with the use of Natural Language Processing techniques.
2. Live agents can be directed into the chatbot when more assistance is required.
3. Collection and verification of personal details such as emails, phone numbers, addresses, and identity numbers.
4. Schedule and arrange delivery dates, meetings, and other appointments.
5. Redirect users to appropriate websites.
6. Direct users to answers via search engines.
7. Perform translation to a different language.
8. Perform sales of goods and collecting payments using payment gateways.

A good use of AI chatbots can immensely benefit businesses in driving sales and shortening the sales cycles.

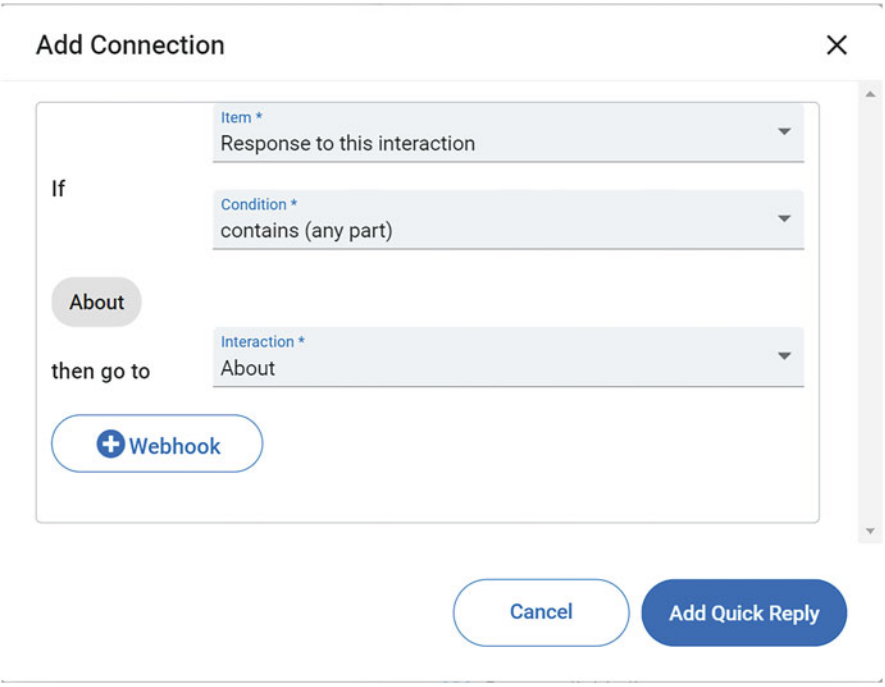



**Fig. 10.11** Add quick reply

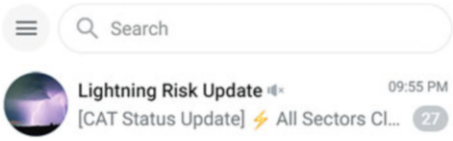



**Fig. 10.12** Telegram search bar

### *10.3.3 Product Recommender Systems*

Recommendation systems are software that analyze a wide range of data collected from customers through online interactions. It can then intelligently analyze each user's usage patterns based on this vast amount of data and provide recommendations for the goods, offers, and bargains that are most likely to appeal to them.

The algorithm powering a recommendation system takes into account things like a person's page views, product interests, time spent on the site, and the types of pages they visit to determine what goods and categories to propose to that user.

Some systems also use language analysis, which enables them to determine which keywords will have the biggest influence when tailoring the shopping experience.

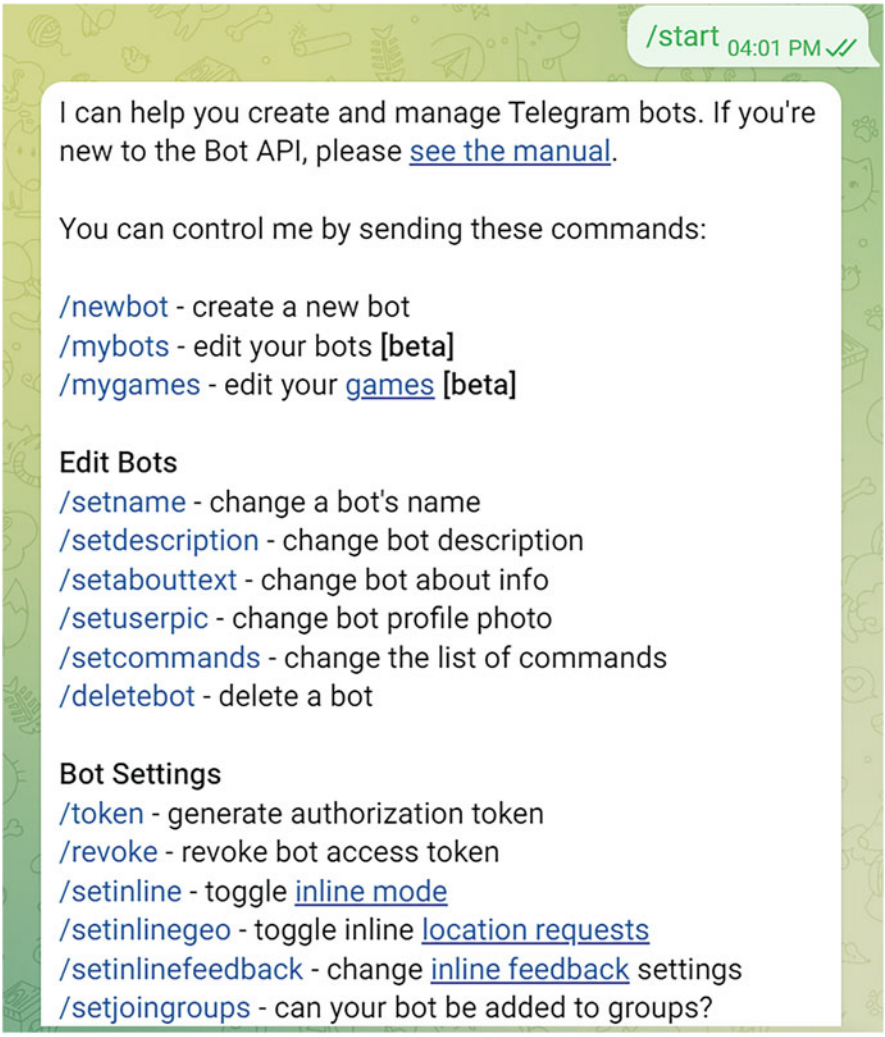



**Fig. 10.13** Create new bot

There are generally 2 main types of recommender systems [7]:

1. **Content filtering**
   This is the quickest recommendation system to create. With a single user, the system may begin to study their activity and generate customized recommendations. It offers personalized recommendations based on the customer's browsing activity.
   The most well-known examples are Netflix and Spotify, which are built on personalization algorithms that assess the content consumption of each visitor in order to recommend films, television shows, and music that correspond to their preferences.

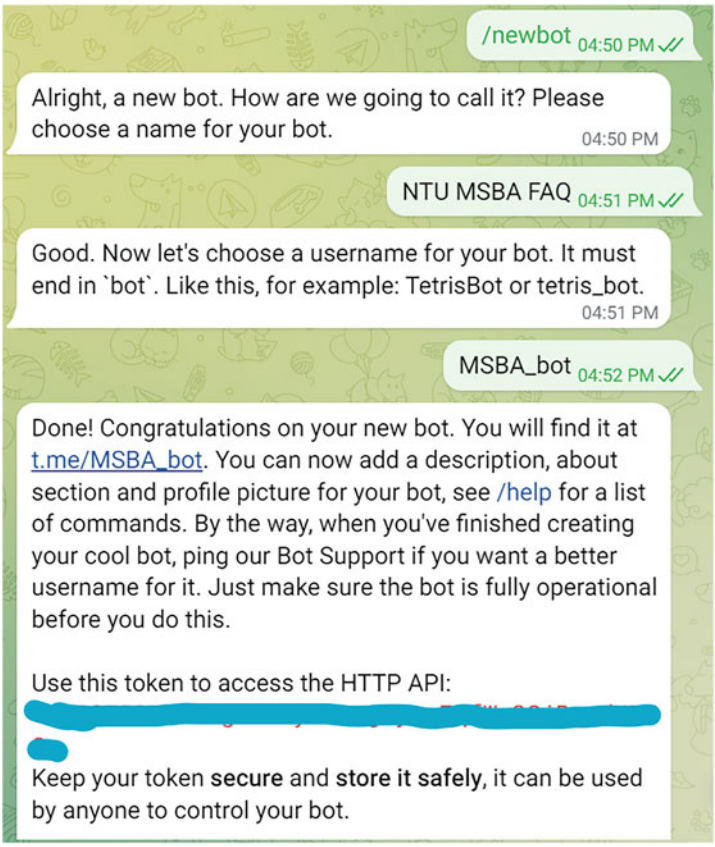



**Fig. 10.14** HTTP API key

Content-based suggestions analyze a customer's previous visits and purchases in order to determine which products best suit their apparent preferences.

The disadvantage of this type of system is that it requires a large amount of constant data input and must evaluate massive amounts of data if the store has a high volume of traffic. This needs several hours of analysis, as well as repeating and revising forecasts whenever new buyers visit or the product catalog is updated.

In addition, for such recommendation systems to be successful and beneficial to the consumer, product content must be well organized and categorized. Only then will the system be able to return the most relevant suggestions.

2. **Collaborative filtering**

Recommendations are generated by providing visitors with suggestions of products or items that have received high ratings from users with comparable characteristics. Instead of producing personalized recommendations based on each user's preferences, the system functions as a bridge between similar

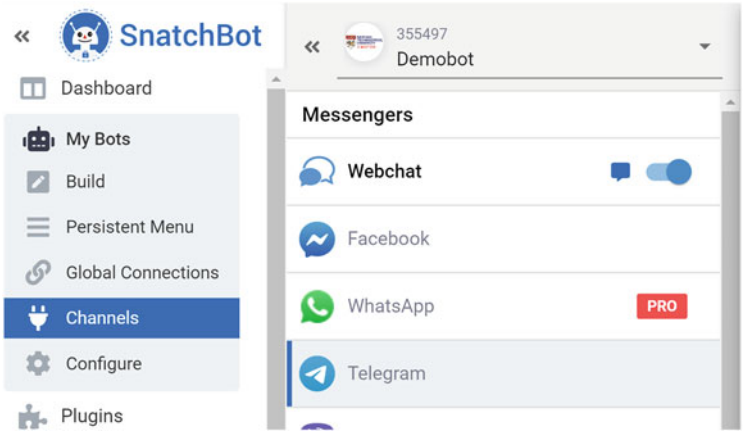



**Fig. 10.15** Set up telegram bot

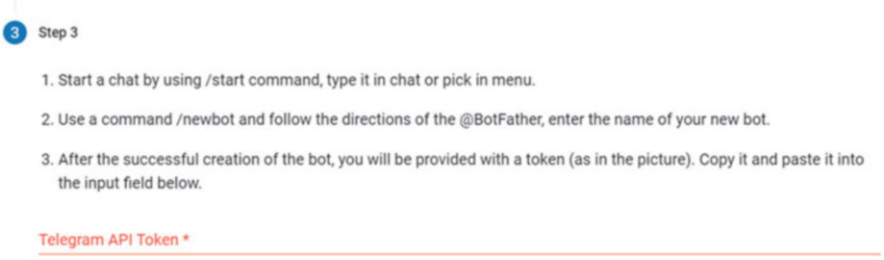



**Fig. 10.16** Fill in HTTP API key

customers. Unlike content filtering, collaborative filtering does not need the items to be categorized. However, it does need a lot of transactions from different users. Amazon popularized this approach, and you will identify it by the "Other customers also bought" section.

Recommender systems—whether they are using content-based, item-based, or user-based filtering methods—all have one requirement in common: their underlying algorithms require a good amount of information for them to generate a relevant product recommendation. When products are new and few customers have purchased them, the amount of information available to recommender systems can be too low to calculate a correlation and the products may not appear within recommendations. This is known as the "cold start." It takes a time for products to warm up and generate enough data to allow recommender systems to produce relevant results.

After attaining sufficient data, we can use AI to analyze objects that are frequently bought together and recommend them to customers who have added one of the items to cart. This will help to increase the number of products sold and boost the revenue of the business.

In this example, we will be creating a recommender system based on collaborative filtering using the Online Retail dataset. The dataset can be obtained from:
https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online Retail.xlsx

```
import pandas as pd
```

First, let us read the data and take a quick look at it.

```
df = pd.read_excel("Sales_Marketing/Online Retail.xlsx
 ↪")
df.head()
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
```

We are only interested in the InvoiceNo, StockCode, Description, and Quantity, so let us filter it out.

```
df_baskets = df[['InvoiceNo', 'StockCode',
 ↪'Description', 'Quantity']]
df_baskets.head()
```

```
  InvoiceNo StockCode                          Description  Quantity
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
```

Now we will convert the dataset into a sparse matrix, listing the items purchased in each transaction using pandas's pivot table.

```
df_items = df_baskets.pivot_table(index='InvoiceNo',␣
 ↪columns=['Description'], values='Quantity').
 ↪fillna(0)
df_items.head(5)
```

```
Description  20713   4 PURPLE FLOCK DINNER CANDLES  \
InvoiceNo
536365         0.0                             0.0
536366         0.0                             0.0
536367         0.0                             0.0
536368         0.0                             0.0
536369         0.0                             0.0

Description   50'S CHRISTMAS GIFT BAG LARGE   DOLLY GIRL BEAKER  \
InvoiceNo
536365                                 0.0                0.0
536366                                 0.0                0.0
536367                                 0.0                0.0
536368                                 0.0                0.0
536369                                 0.0                0.0

Description   I LOVE LONDON MINI BACKPACK   I LOVE LONDON MINI RUCKSACK  \
InvoiceNo
536365                               0.0                           0.0
536366                               0.0                           0.0
536367                               0.0                           0.0
536368                               0.0                           0.0
536369                               0.0                           0.0

Description   NINE DRAWER OFFICE TIDY   OVAL WALL MIRROR DIAMANTE   \
InvoiceNo
536365                            0.0                          0.0
536366                            0.0                          0.0
536367                            0.0                          0.0
536368                            0.0                          0.0
536369                            0.0                          0.0

Description   RED SPOT GIFT BAG LARGE   SET 2 TEA TOWELS I LOVE LONDON   ...  \
InvoiceNo                                                                 ...
536365                            0.0                                0.0  ...
536366                            0.0                                0.0  ...
536367                            0.0                                0.0  ...
536368                            0.0                                0.0  ...
536369                            0.0                                0.0  ...

Description  wrongly coded 20713  wrongly coded 23343  wrongly coded-23343  \
InvoiceNo
536365                       0.0                  0.0                  0.0
536366                       0.0                  0.0                  0.0
536367                       0.0                  0.0                  0.0
536368                       0.0                  0.0                  0.0
536369                       0.0                  0.0                  0.0

Description  wrongly marked  wrongly marked 23343  \
InvoiceNo
536365                 0.0                   0.0
536366                 0.0                   0.0
536367                 0.0                   0.0
536368                 0.0                   0.0
536369                 0.0                   0.0

Description  wrongly marked carton 22804  wrongly marked. 23343 in box  \
InvoiceNo
```

(continues on next page)

(continued from previous page)

```
536365                                0.0                                0.0
536366                                0.0                                0.0
536367                                0.0                                0.0
536368                                0.0                                0.0
536369                                0.0                                0.0

Description  wrongly sold (22719) barcode  wrongly sold as sets  \
InvoiceNo
536365                                 0.0                   0.0
536366                                 0.0                   0.0
536367                                 0.0                   0.0
536368                                 0.0                   0.0
536369                                 0.0                   0.0

Description  wrongly sold sets
InvoiceNo
536365                     0.0
536366                     0.0
536367                     0.0
536368                     0.0
536369                     0.0

[5 rows x 4223 columns]
```

### *10.3.4 Recommending via Pairwise Correlated Purchases*

One simple way to recommend products frequently purchased together is to use correlation to identify objects which are often purchased together in the same transaction. When two objects are frequently purchased together, the correlation score would be high. On the other hand, the correlation score would be low if they are not frequently purchased together. In this example, we can further filter results to have a minimum correlation of 0.3 to filter out weak correlations.

```
def get_recommendations(df, item, threshold=0.3):
    """Generate a set of product recommendations␣
↪using item-based collaborative filtering.

    Args:
        df (dataframe): Pandas dataframe containing␣
↪matrix of items purchased.
        item (string): Column name for target item.

    Returns:
        recommendations (dataframe): Pandas dataframe␣
↪containing product recommendations.
    """
```

(continues on next page)

(continued from previous page)

```
    recommendations = df.corrwith(df[item])
    recommendations.dropna(inplace=True)
    recommendations = pd.DataFrame(recommendations,␣
↪columns=['correlation']).reset_index()
    recommendations = recommendations.sort_values(by=
↪'correlation', ascending=False)
    return recommendations[recommendations[
↪"correlation"] > threshold]
```

By querying for the WHITE HANGING HEART T-LIGHT HOLDER, we can see that GIN + TONIC DIET METAL SIGN and FAIRY CAKE FLANNEL ASSORTED COLOUR have a high correlation score of 0.8 and would make a great recommendation.

```
recommendations = get_recommendations(df_items,
↪'WHITE HANGING HEART T-LIGHT HOLDER')
recommendations
```

```
                            Description  correlation
3918  WHITE HANGING HEART T-LIGHT HOLDER     1.000000
1478         GIN + TONIC DIET METAL SIGN     0.824987
1241  FAIRY CAKE FLANNEL ASSORTED COLOUR     0.820905
1072                 DOORMAT FAIRY CAKE     0.483524
3627              TEA TIME PARTY BUNTING     0.469207
2847    RED HANGING HEART T-LIGHT HOLDER     0.342551
3630               TEA TIME TEA TOWELS      0.337001
1710              HEART OF WICKER SMALL     0.311869
```

However, using correlations in this manner will only tell us about the relationships between two items. In order to find a third recommendation when pairs of items were purchased together, we will need to group the itemsets together and there will be many pairings. Another way we can identify prominent recommendations is by association rule mining.

Association rules are "if-then" statements, which help to show the probability of relationships between data items, within large data sets in various types of databases.

Association rule mining is the process of engineering data into a predictive feature in order to fit the requirements or to improve the performance of a machine learning model. The Apriori Algorithm is an algorithm used to perform association rule mining over a structured dataset.

An itemset is a set of items in a transaction. An itemset of size 3 means there are 3 items in the set. In general, it can be of any size, unless specified otherwise.

Association Rule

- Forms: X => Y.
- X associated with Y.

- X is the “antecedent” itemset, and Y is the “consequent” itemset.
- There might be more than one item in X or Y.

In order to select the interesting rules out of multiple possible rules from this small business scenario, we will be using the following measures:

- Support
  Support is an indication of how frequently the item appears in the dataset. For example, how popular a product is in a shop.

```
The support for the combination A and B would be,
    - P(AB) or P(A) for Individual A
```

- Confidence
  Confidence is an indication of how often the rule has been found to be true. It indicates how reliable the rule is. For example, how likely is it that someone would buy toothpaste when buying a toothbrush.

```
In other words, confidence is the conditional␣
↪probability of the consequent given the antecedent,
    - P(B|A), where P(B|A) = P(AB)/P(A)
```

- Lift
  Lift is a metric to measure the ratio of the confidence of products occurring together if they were statistically independent. For example, how likely is another product purchased when purchasing a product, while controlling how popular the other product is.
  A lift score that is smaller than 1 indicates that the antecedent and the consequent are substitutes of each other. This means that the occurence of the antecedent has a negative impact on the occurence of the consequent. A lift score that is greater than 1 indicates that the antecedent and consequent are dependent to each other, and the occurrence of antecedent has a positive impact on the occurrence of consequent. A lift score that is smaller than 1 indicates that the antecedent and the consequent are substitute each other that means the existence of antecedent has a negative impact to consequent or vice versa.

```
Consider an association rule ``if A then B.” The␣
↪lift for the rule is defined as
  - P(B|A)/P(B), which is also P(AB)/(P(A)*P(B)).
As shown in the formula, lift is symmetric in that␣
↪the lift for ``if A then B” is the same as the lift␣
↪for ``if B then A.”
```

**Fig. 10.17** Example transaction database

Example database with 5 transactions and 5 items

| transaction ID | milk | bread | butter | beer | diapers |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 |

- Conviction
  Conviction score is a ratio between the probability that one product occurs without another while they were dependent and the actual probability of one product's existence without another. It measures the implied strength of the rule from statistical independence, for example, if the (oranges) → (apples) association has a conviction score of 1.5; the rule would be incorrect 1.5 times more often (50% more often) if the association between the two were totally independent.

```
The Conviction score of A -> B would be defined as :
  - (1 - Support(B)) / (1 - Confidence(A -> B))
```

Using this small example in Fig. 10.17, we can calculate the support, confidence, and lift of a rule.

For example, for the rule {milk, bread} => Butter, we can calculate the following measures:

- Support ({milk}) = 2/5 = 0.4
- Support ({milk, bread}) = 2/5 = 0.4
- Confidence ({milk, bread} => Butter)= 1/2 = 0.5
- Lift ({milk, bread} => Butter) = (1/2) / (2/5) = 1.25

We can find multiple rules from this scenario. For example, in a transaction of milk and bread, if milk is bought, then customers also buy bread.

In the following example, we will mine for good association rules to identify good recommendations by analyzing the data.

```
import numpy as np
df_associations = (df_items >= 1).astype(np.int)
```

We will use a small support of 0.02 as the data is sparse since there are many transactions and many item options for our customers to purchase. A confidence threshold of 0.5 is set so that we select good associations.

```
from mlxtend.frequent_patterns import apriori,␣
↪association_rules
item = apriori(df_associations, use_colnames=True,␣
↪min_support=0.02)
item.head()
```

```
    support                            itemsets
0  0.039066             (6 RIBBONS RUSTIC CHARM)
1  0.025280  (60 CAKE CASES VINTAGE CHRISTMAS)
2  0.033871       (60 TEATIME FAIRY CAKE CASES)
3  0.025157   (72 SWEETHEART FAIRY CAKE CASES)
4  0.040088        (ALARM CLOCK BAKELIKE GREEN)
```

```
rules = association_rules(item, metric = 'confidence',
↪ min_threshold=0.5)
rules.head()
```

```
                  antecedents                  consequents  \
0    (ALARM CLOCK BAKELIKE RED )   (ALARM CLOCK BAKELIKE GREEN)
1   (ALARM CLOCK BAKELIKE GREEN)    (ALARM CLOCK BAKELIKE RED )
2  (CHARLOTTE BAG PINK POLKADOT)  (RED RETROSPOT CHARLOTTE BAG)
3  (RED RETROSPOT CHARLOTTE BAG)  (CHARLOTTE BAG PINK POLKADOT)
4    (CHARLOTTE BAG SUKI DESIGN)  (RED RETROSPOT CHARLOTTE BAG)

   antecedent support  consequent support   support  confidence       lift  \
0            0.042993             0.040088  0.026180    0.608944  15.190043
1            0.040088             0.042993  0.026180    0.653061  15.190043
2            0.030394             0.042297  0.021353    0.702557  16.609974
3            0.042297             0.030394  0.021353    0.504836  16.609974
4            0.036080             0.042297  0.020740    0.574830  13.590225

   leverage  conviction
0  0.024457    2.454665
1  0.024457    2.758433
2  0.020068    3.219788
3  0.020068    1.958151
4  0.019214    2.252517
```

From the earlier analysis, we can see that people who purchase ALARM CLOCK BAKELIKE RED tend to also purchase ALARM CLOCK BAKELIKE GREEN. Thus, it might be meaningful to consider creating discounted variety packs for alarm clocks to encourage more purchases.

Additionally, we are able to recommend other similar products to help the customer pick better products across relevant searches. This could help raise customer satisfaction by allowing them to products that would better suit their needs.

In this example, we will implement a text-based search engine

```
import pandas as pd
df = pd.read_excel("Sales_Marketing/Online Retail.xlsx
 ↪")
```

In this example, we will search for products with descriptions similar to the "WHITE HANGING HEART T-LIGHT HOLDER." To do this, we will convert all the descriptions found in the dataset into a document embedding using the flair library and compare it against the document embedding of "WHITE HANGING HEART T-LIGHT HOLDER."

```
from flair.Embeddings import WordEmbeddings,␣
 ↪DocumentPoolEmbeddings
from flair.data import Sentence
import torch

# initialize the word embeddings
glove_embedding = WordEmbeddings('glove')

# initialize the document embeddings, mode = mean
document_embeddings = DocumentPoolEmbeddings([glove_
 ↪embedding])
```

First, we need to convert all the descriptions in the dataset to embeddings.

```
product_description = df["Description"].dropna().
 ↪unique()

Embeddings = []
for description in product_description:
    # create an example sentence
    sentence = Sentence(str(description))
    # embed the sentence with our document embedding
    document_embeddings.embed(sentence)
    Embeddings.append(sentence.embedding)

Embeddings = torch.stack(Embeddings)
```

Now let us get the document embedding of "WHITE HANGING HEART T-LIGHT HOLDER." We will use cosine similarity to identify the top 5 most similar descriptions. In this case, we can see that the "RED HANGING HEART T-LIGHT HOLDER" and "PINK HANGING HEART T-LIGHT HOLDER" are other possible relevant items that the user could be interested in.

```
# create an example sentence
sentence = Sentence('WHITE HANGING HEART T-LIGHT␣
↪HOLDER')

# embed the sentence with our document embedding
document_embeddings.embed(sentence)

cos = torch.nn.CosineSimilarity(dim=1)
output = cos(sentence.embedding.unsqueeze(0),␣
↪Embeddings)
output.argsort(descending=True)
product_description[output.
↪argsort(descending=True)[1:6].cpu()]
```

```
array(['RED HANGING HEART T-LIGHT HOLDER',
       'PINK HANGING HEART T-LIGHT HOLDER', 'BLACK␣
↪HEART CARD HOLDER',
       'CREAM HANGING HEART T-LIGHT HOLDER',
       'HEART STRING MEMO HOLDER HANGING'],␣
↪dtype=object)
```

Through various uses such as lead scoring, recommendation systems, and chatbots, artificial intelligence is able to help streamline and optimize sales processes, bringing in increased productivity and profit for businesses.

## Exercises

1. List the different roles and responsibilities of the sales team.
2. For each stage of the sales cycle below, explain how making improvements to each stage will impact the sales process.
   - Prospecting
   - Making contact
   - Qualifying leads
   - Lead nurturing
   - Making an offer
   - Handling objections
   - Closing a sale
   - Delivering
3. List three different challenges that sales departments face.
4. For each of the challenges listed in 2, identify one way that artificial intelligence could potentially be used to alleviate the problem.

5. Develop a chatbot that helps answer customer queries and direct the flow of conversation to different sales teams.

   Assume that you work in a business that sells beauty products. The company sells skincare, body wash, shampoo, and makeup products targeted for people of different age groups and gender.

   In this exercise, try to use what you have learned so far to develop a chatbot that can:

   - Direct potential buyers to the relevant sales team.
   - Help answer frequently asked questions to potential customers.
   - Schedule meetings with salespeople.

## References

1. Alagh H (2022) 7 Ways to Shorten the Sales Cycle Using Customer Experience Automation. https://yellow.ai/customer-experience/enhance-sales-cycle-with-cx-automation/. Accessed 27 Apr 2023
2. Ashydv (2019) Leads dataset. In: Kaggle. https://www.kaggle.com/datasets/ashydv/leads-dataset. Accessed 27 Apr 2023
3. Brian A (2020) How Artificial Intelligence in Sales is Changing the Selling Process. https://www.ai-bees.io/post/how-artificial-intelligence-in-sales-is-changing-the-selling-process. Accessed 27 Apr 2023
4. Brudner E (2021) Salespeople Only Spent One-Third of Their Time Selling Last Year [Infographic]. https://blog.hubspot.com/sales/salespeople-only-spent-one-third-of-their-time-selling-last-year. Accessed 27 Apr 2023
5. James O, Kristina M, David E (2011) Harvard Business Review. https://hbr.org/2011/03/the-short-life-of-online-sales-leads. Accessed 27 Apr 2023
6. Mackenzie (2020) What is Lead Scoring? and Why It's Important to Sales Success. https://evenbound.com/blog/what-is-lead-scoring. Accessed 27 Apr 2023
7. Muñoz A (2021) How Recommendation Systems Can Boost Sales | Sales Layer. https://blog.saleslayer.com/recommendation-systems-ecommerce. Accessed 27 Apr 2023

# Chapter 11
# AI in Marketing


**Learning Outcomes**

- Understand the roles and responsibilities of marketing departments.
- Be able to explain the differences in roles between sales and marketing teams.
- Identify ways that artificial intelligence can be applied to marketing.
- Develop artificial intelligence solutions for customer segmentation and identify word which brands are associated with.

## 11.1 Introduction to Marketing

Marketing is the process of attracting potential clients to a company's products or services. The overarching purpose of marketing is to persuade customers to buy a product or service by communicating the benefits it can bring. Market research helps businesses determine their target audience and understand their needs. Based on their research, marketing teams will develop a marketing strategy to formulate the 4Ps of marketing: Product, Pricing, Promotion, and Place.

The marketing department is tasked with cultivating interest for the company's products and is also responsible for creating and maintaining the image of a company. The scope of marketing teams includes:

1. **Brand image and perception**
   Brand image is the customer's impression about a brand and is developed over time. Customers' perceptions of a brand are formed based on their experiences and interactions with the brand. These interactions can occur in a variety of ways, not just through the purchase or use of a product or service.
   When a consumer makes a purchase, he or she is also purchasing the image associated with the product or service offered. Therefore, brands need to be distinctive, positive, and immediately recognizable. Brand communication, such



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_11

as product packaging, advertisements, word-of-mouth marketing, and other forms of promotion, can also be used to bolster the brand's reputation.

2. **Market research**
   Market research provides critical information essential to understand your market and competitive landscape. Knowing how your target audience views your business is important in expanding your business. It can provide insights to better engage with them, analyze our performance compared to the competition, and highlight the way forward. All these insights will lead to better product development, return on advertising, customer satisfaction, and lead generation.
3. **Developing and disseminating promotional content**
   Content marketing provides answers to audience inquiries while also fostering relationships, enhancing conversions, and generating leads. Customers are more likely to do business with a company if they feel confident that they are dealing with industry experts. Thus, content marketing is a great way to prove your brand's knowledge and credibility in your niche. Content marketing can help build trust in your brand while imparting useful information that will enable the reader to make a wiser choice when making a purchase.
4. **Tracking trends and monitoring competition campaigns**
   Keeping up with the latest trends helps to generate more original ideas and knowing which trends are popular might help you produce similar content and remain relevant. By monitoring and analyzing these trends, marketing teams can identify ways to improve, reach, and engage the target audience better. To stay ahead of the competition, it is important to pay attention to the latest industry trends and find out how to implement them into your own firm.
   Understanding the competition will enable businesses to better anticipate the innovations of rivals and strategize how to respond to them. Keeping an eye on the competition can also help avoid making poor business choices. Furthermore, we can avoid making mistakes by learning from their failures.
5. **Performing optimization of marketing campaigns**
   Improving marketing campaigns can be done in a variety of ways such as working on search engine optimization, improving content scores, identifying better target audiences, etc. Through search engine optimization, we can improve online traffic and increase the reach of products and services to a greater number of people. This will allow us to generate more leads which will ideally increase sales.
   Content scoring enables us to evaluate and measure the potential of a piece of content by monitoring the performance of prior individual content pieces in terms of lead generation and conversion. Improving the quality of content is pivotal to the success of marketing campaigns and can help improve engagement with the audience.

### *11.1.1 Sales vs Marketing*

Sales and marketing teams seem to have many similar goals, such as getting more leads, turning those leads into new customers, and making as much money as possible. While these two departments must cooperate to successfully achieve their objectives, there are a few key differences.

1. Objectives: Marketing focuses on generating interest in your business's products or services, while sales teams aim to convincing potential customers to make their purchases.
2. Methods: Sales teams generally contact leads using more personal interactions such as trade show networking, face-to-face meetings, cold calling, and retail interaction. Marketing strategies typically utilize a more impersonal approach, such as print and TV advertising, social media marketing, email marketing, search engine optimization (SEO), and digital marketing.
3. Scope: Sales performance indicators are oriented toward hitting short-term quotas, while marketing objectives usually focus on long-term big-picture targets.
4. Responsibilities: Sales teams are responsible for selling existing products and services. Whereas marketing teams may be included in the product development process through providing insights gained from market research and translating that into product properties which would appeal to a target demographic.
5. Tools: Sales departments manage the sales cycle, organize communication with leads, and prioritize tasks through the use of CRM software (customer relationship management). On the other hand, marketing departments mainly utilize marketing automation software to manage their digital and email marketing campaigns, as well as track marketing-qualified leads.

## 11.2 Artificial Intelligence in Marketing

These days, artificial intelligence and machine learning are at the forefront of the advertising world. This is primarily due to the fact that such technologies provide an opportunity to transform marketing tools and datasets into valuable marketing insights, which would help a company increase its return on investment. When it comes to content marketing, a company risks losing its edge in the market if it does not implement marketing solutions using artificial intelligence.

AI enables organizations to extract insights from online activity about their followers, which helps them to target more relevant content and advertisements to the appropriate audiences. Furthermore, machine learning and natural language processing can help us gather insights about what customers are looking for and analyze the content they have posted on their social media profiles.

The use of AI may also enable marketing teams to reply quickly to these clients while still being relevant and helpful though suggested responses. Blog posting and content creation can also be assisted by AI though generating ideas and creating

initial drafts for marketing teams to vet through and improve on. It can also analyze consumer behavior, engagement, and sentiment to assist future marketing decisions. By responding to consumer posts in real time and tailoring its marketing to those customers, the incorporation of AI into marketing has the potential to provide more customized experiences for the company's customers.

Ways that AI is Changing Marketing:

1. Lead Generation
2. Customer Segmentation
3. Content Creation
4. Personalized website experiences
5. Competition Analysis
6. Brand Analysis
7. Market Basket Analysis

## 11.3 Applications of AI in Marketing

### *11.3.1 Customer Segmentation*

Customer segmentation is the process of splitting customers into smaller groups which contain similar kinds of people for the purpose of marketing.

Customer segmentation operates under the premise that marketing efforts would be better served if they targeted particular, smaller groups with messages relevant to those customers and encourage them to purchase something.

Additionally, companies also hope to obtain a deeper understanding of their consumers' tastes and demands, so as to determine what each client segment values most. This information would help to target their marketing materials more precisely to the relevant groups.

When determining customer segmentation practices, many factors are considered such as demographic, geographic, psychographic, and behavioral attributes.

The dataset we will be using can be obtained from the link [3]:

https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python

This dataset contains information about the customers who visit a specific supermarket mall. Information about their spending habits is collected from membership card sign ups and spending habits. The spending score is assigned to customers based on known customer behaviors according to their preferences and purchasing data. Our task here is to provide the marketing team with insights about the different customer groups and characterize their spending habits for targeted advertising campaigns.

```
import os, warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,␣
↪MinMaxScaler
from scipy.cluster import hierarchy
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

cust = pd.read_csv('Sales_Marketing/Mall_Customers.csv
↪', sep=',')
cust.rename(columns={"Annual Income (k$)": "Annual␣
↪Income", "Spending Score (1-100)": "Spending Score"}
↪, inplace=True)
print("There are {:,} observations and {} columns in␣
↪the data set.".format(cust.shape[0], cust.shape[1]))
print("There are {} missing values in the data.".
↪format(cust.isna().sum().sum()))
cust.head()
```

```
There are 200 observations and 5 columns in the data set.
There are 0 missing values in the data.
```

| | CustomerID | Gender | Age | Annual Income | Spending Score |
|---|---|---|---|---|---|
| 0 | 1 | Male | 19 | 15 | 39 |
| 1 | 2 | Male | 21 | 15 | 81 |
| 2 | 3 | Female | 20 | 16 | 6 |
| 3 | 4 | Female | 23 | 16 | 77 |
| 4 | 5 | Female | 31 | 17 | 40 |

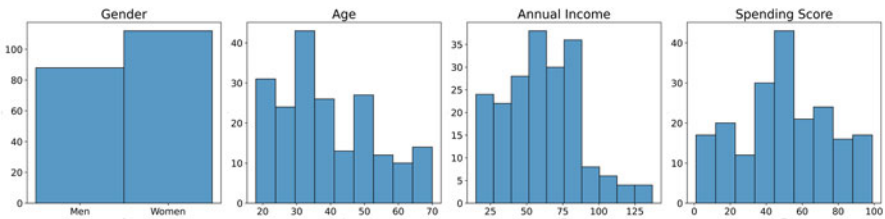
```
cust.drop('CustomerID', axis=1, inplace=True)
pd.DataFrame(cust.describe()).style.set_caption(
↪"Summary Statistics of Numeric Variables")
```

```
<pandas.io.formats.style.Styler at 0x1813d5da820>
```

```
cust['Gender'] = ['Women' if i == 'Female' else 'Men'␣
↪for i in cust.Gender]
pd.DataFrame(cust.select_dtypes('object').describe().
↪T).style.set_caption("Summary Statistics of␣
↪Categorical Variables")
```

```
<pandas.io.formats.style.Styler at 0x1815edc7eb0>
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a subplot with multiple plots
fig, axs = plt.subplots(1, 4, figsize=(20, 5))
axs = axs.ravel()

# Plot histograms for all columns
for i, column in enumerate(cust.columns):
    sns.histplot(cust[column], ax=axs[i])
    axs[i].set_title(column)

plt.tight_layout()
plt.show()
```




```
# K-Means Clustering
clust_df = cust.copy()
clust_df['Gender'] = [1 if i == "Women" else 0 for i in clust_df.Gender]

k_means = list()
max_silhouette_score = -np.inf
best_cluster_number = 2
for n_clusters in range(2,10):
    km = KMeans(n_clusters=n_clusters, init='k-means++', random_state=21).
↪fit(clust_df)
    kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=100)
    kmeans.fit(clust_df)
    clusters_customers = kmeans.predict(clust_df)
```

(continues on next page)

(continued from previous page)

```
    silhouette_avg = silhouette_score(clust_df, clusters_customers)
    print('number of clusters: {}, silhouette score: {:<.3f}'.format(n_
↪clusters, silhouette_avg))

    if silhouette_avg > max_silhouette_score:
        best_cluster_number = n_clusters
        max_silhouette_score = silhouette_avg
print("Best silhouette score: {0}, Recommended number of clusters: {1}".
↪format(max_silhouette_score, best_cluster_number))
```

```
number of clusters: 2, silhouette score: 0.293
number of clusters: 3, silhouette score: 0.384
number of clusters: 4, silhouette score: 0.405
number of clusters: 5, silhouette score: 0.444
number of clusters: 6, silhouette score: 0.452
number of clusters: 7, silhouette score: 0.441
number of clusters: 8, silhouette score: 0.428
number of clusters: 9, silhouette score: 0.414
Best silhouette score: 0.45205475380756527, Recommended number of clusters: 6
```

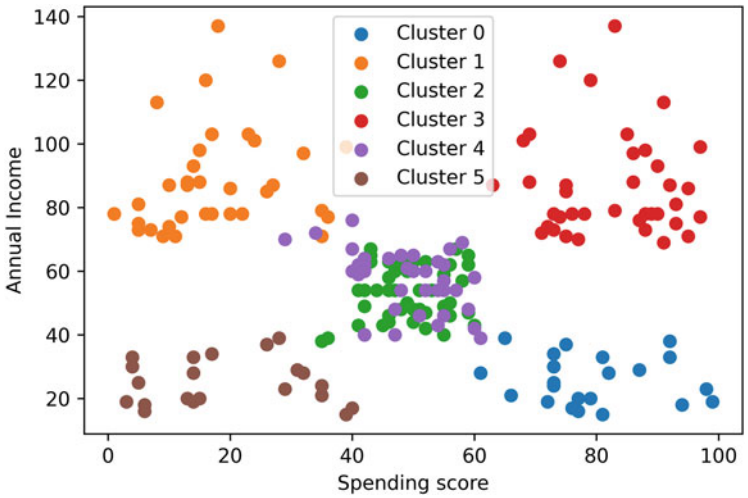
It seems like the algorithm recommends 6 clusters based on the silhouette score. Let us plot our clusters with respect to the original features to visualize how it is clustering the data.

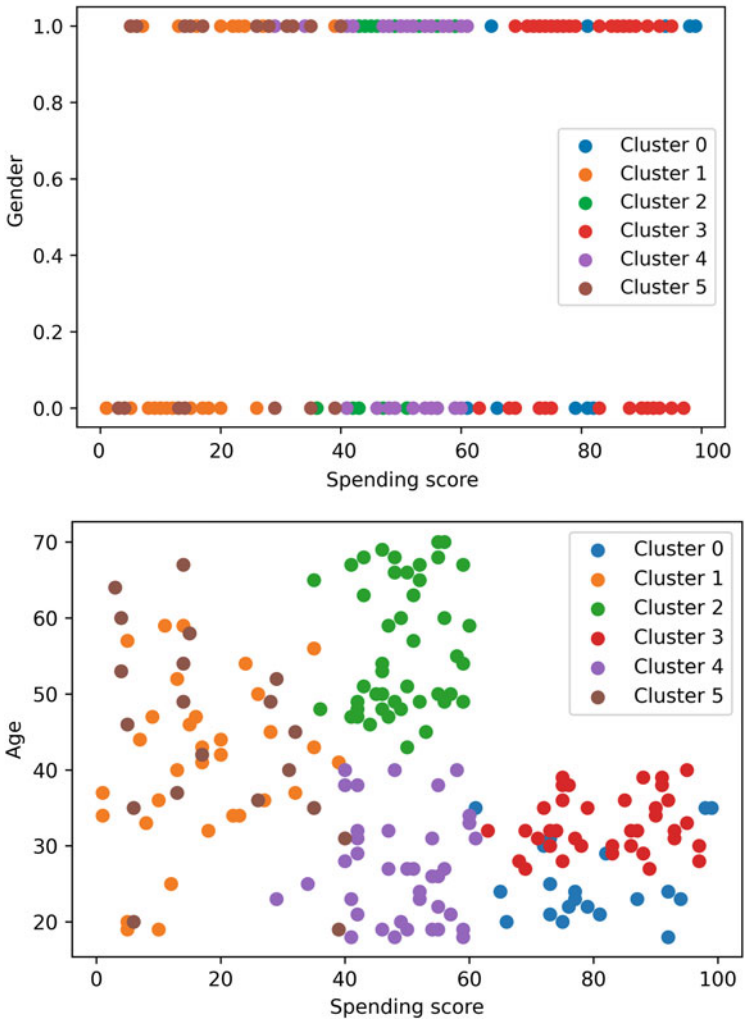
```
km = KMeans(n_clusters=best_cluster_number, init='k-
↪means++', random_state=21).fit(clust_df)
km_pred = km.fit_predict(clust_df)
plot_km=clust_df.copy()
plot_km['K-Means Cluster'] = km_pred
plot_km=plot_km.sort_values(by='K-Means Cluster')

# Plot of clusters
plt.figure(0)
for cluster_id in range(best_cluster_number):
    plt.scatter(plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Spending Score"],
                plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Annual Income"])
plt.legend(["Cluster {0}".format(i) for i in␣
↪range(best_cluster_number)])
plt.xlabel("Spending score")
plt.ylabel("Annual Income")
plt.figure(1)
for cluster_id in range(best_cluster_number):
    plt.scatter(plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Spending Score"],
                plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Gender"])
```

(continues on next page)

(continued from previous page)

```
plt.legend(["Cluster {0}".format(i) for i in␣
↪range(best_cluster_number)])
plt.xlabel("Spending score")
plt.ylabel("Gender")
plt.figure(2)
for cluster_id in range(best_cluster_number):
    plt.scatter(plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Spending Score"],
                plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Age"])
plt.legend(["Cluster {0}".format(i) for i in␣
↪range(best_cluster_number)])
plt.xlabel("Spending score")
plt.ylabel("Age")
plt.show()
```





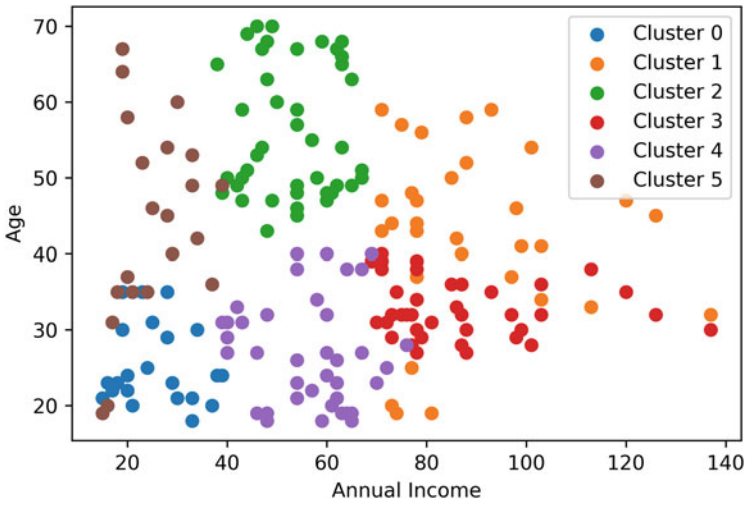

It seems that 6 clusters express the data well. 5 clusters seem distinct when using the Annual income attribute, but when looking at the Age attribute, the center cluster of clusters 2 and 4 could be split further into 2 more distinct clusters. Let us plot the two attributes out.

```
for cluster_id in range(best_cluster_number):
    plt.scatter(plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Annual Income"],
                plot_km[plot_km["K-Means Cluster"] ==␣
↪cluster_id]["Age"])
```

(continues on next page)

(continued from previous page)

```
plt.legend(["Cluster {0}".format(i) for i in␣
 ↪range(best_cluster_number)])
plt.xlabel("Annual Income")
plt.ylabel("Age")
plt.show()
```




Thus we can cluster our customers into 6 types:

Cluster 0: young and low income (high spending score)
Cluster 1: old and high income (low spending score)
Cluster 2: old and middle income (average spending score)
Cluster 3: young and high income (high spending score)
Cluster 4: young and middle income (average spending score)
Cluster 5: old and low income (low spending score)

In this example, we have demonstrated how to perform customer segmentation based on their attributes. Through this, we are able to better profile the different types of customers and characterize their spending habits for marketing teams to understand their audience profiles better and create better targeted advertisements. For example, we could create promotion campaigns for daily needs to entice specifically older people to visit by aiming for volume since they spend lesser. On the other hand, we could focus on promoting lifestyle products to younger customers that may be more expensive but attractive as they are more willing to spend.

### *11.3.2 Analyzing Brand Associations*

In this example, we will try to identify words that people associate with a particular brand in comparison to others. This helps to identify differentiating features that make up the brand identity and also helps marketers understand how effective their campaigns for brand awareness are. We will achieve this by training a Logistic Regression model to differentiate brands from one another and identify the words that carry the most weight to identify a particular brand.

```
from sklearn.feature_extraction.text import␣
↪TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,␣
↪balanced_accuracy_score
from sklearn.feature_extraction.text import␣
↪CountVectorizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

Let us import the dataset.

```
reviews = pd.read_csv("Sales_Marketing/car_5_brands.
↪csv")
reviews.head()
```

```
   Unnamed: 0  Rating  car_year brand_name        date  \
0           0     5.0      2018       Audi  2018-07-11
1           1     5.0      2018       Audi  2018-06-24
2           2     5.0      2018       Audi  2018-05-02
3           3     5.0      2018       Audi  2017-12-07
4           4     5.0      2018       Audi  2017-10-25

                                              review
0  BEST ALL AROUND PURPOSE CROSSOVER SUV I have n...
1  Best car This is a wonderful car.  The technol...
2                        Great Buy Do your home work
3  Fun Car Great ride. Loaded with technology. St...
4  Best luxury SUV w/ perfect comfort/sport balan...
```

Now we will prepare the brand labels by mapping brands to integer classes.

```
label_map = {brand: idx for idx, brand in␣
↪enumerate(reviews["brand_name"].unique())}
reviews["brand_labels"] = reviews["brand_name"].
↪map(label_map)
```

We will next drop all empty reviews.

```
reviews.dropna(subset=['review'], inplace=True)
```

Drop all tokens with numbers in them as they usually do not describe a brand well.

```
reviews["processed_text"] = reviews["review"].
↪apply(lambda row: " ".join([word for word in row.
↪split(" ") if word.isalpha()]))
```

We will also predict and drop nouns as we are interested in words that can define a brand apart from its products and services.

```
reviews["processed_text"] = reviews["processed_text"].
↪apply(lambda row: " ".join([i[0] for i in nltk.tag.
↪pos_tag(word_tokenize(row)) if not i[1].startswith(
↪'NN')]))
```

Now we will need to remove the stopwords, brand_name, lowercase the words, and use CountVectorizer to count the occurrence of each word. Then we can split the data into the training and testing sets.

```
docs=reviews["processed_text"].tolist()
stop_words = stopwords.words('english')
[stop_words.append(i.lower()) for i in reviews["brand_
↪name"].unique()]

cv=CountVectorizer(max_df=0.90, lowercase=True, stop_
↪words=stop_words)
word_count_vector=cv.fit_transform(docs)

X_train, X_test, y_train, y_test = train_test_
↪split(word_count_vector, reviews["brand_labels"],␣
↪test_size=0.2, random_state=42)
```

Let us check the class counts.

```
np.unique(y_train, return_counts=True)
```

```
(array([0, 1, 2, 3, 4], dtype=int64),
 array([4839, 4895, 3436, 6454, 5926], dtype=int64))
```

It looks relatively balanced. Now we will apply TF-IDF to extract the importance of the terms and train the model.

```
tfidf_transformer=TfidfTransformer(smooth_idf=True,
 ↪use_idf=True)
tfidf_x_train = tfidf_transformer.fit_transform(X_
 ↪train)
model1 = LogisticRegression(max_iter=1000)
model1.fit(tfidf_x_train, y_train)
tfidf_x_test = tfidf_transformer.transform(X_test)
y_pred = model1.predict(tfidf_x_test)
cm=confusion_matrix(y_test, y_pred)
acc=balanced_accuracy_score(y_test, y_pred)
print('Accuracy of a simple linear model with TFIDF␣
 ↪is .... {:.2f}%'.format(acc*100))
```

```
Accuracy of a simple linear model with TF-IDF is ....␣
 ↪29.52%
```

It looks like we can classify the brand with 30% accuracy, which is higher than 20% for 5 brands. Keep in mind that we have removed most of the easily differentiating words such as brand-specific products and items. Now let us see which words are more correlated with a particular brand.

```
for brand, brand_idx in label_map.items():
    idx = model1.coef_[brand_idx].argsort()[::-
 ↪1][0:25]
    print(brand, np.array(cv.get_feature_
 ↪names())[idx])
    print()
```

```
Audi ['quattro' 'turbo' 'avant' 'german' 'tiptronic' 'timing' 'tie'
 'understated' 'wheel' 'snowy' 'virtual' 'supercharged' 'glued'
 'headlight' 'manual' 'quatro' 'settled' 'silly' 'magnetic' 'premium'
 'automotive' 'wait' 'subtle' 'quart' 'given']

Lexus ['hybrid' 'quiet' 'hesitates' 'ls' 'smooth' 'older' 'rides' 'floating'
 'luxurious' 'present' 'advised' 'remote' 'smoothest' 'complained' 'gs'
 'reliable' 'leasing' 'negative' 'pebble' 'listed' 'soft' 'dropped'
 'forever' 'annoying' 'suitable']

INFINITI ['tech' 'intelligent' 'searched' 'airbag' 'watching' 'dvd' 'uneven'
 'touring' 'spacious' 'dependable' 'bring' 'catalytic' 'falling' 'held'
 'reasonable' 'owed' 'poor' 'secret' 'trade' 'fight' 'leased' 'stand'
 'confidently' 'rotate' 'admiring']

BMW ['idrive' 'ultimate' 'xdrive' 'flat' 'convertible' 'steptronic' 'hugs'
 'manual' 'active' 'wow' 'provide' 'throttle' 'assembly' 'classic'
 'polarized' 'window' 'cup' 'master' 'bavarian' 'ordered' 'cooling' 'cold'
 'balanced' 'handled' 'automatic']
```

(continues on next page)

(continued from previous page)

```
Mercedes-Benz ['mercedes' 'amg' 'airmatic' 'matic' 'retractable' 'electrical
↪' 'ml'
 'benz' 'diesel' 'tall' 'mb' 'solid' 'fails' 'raise' 'classic' 'clk'
 'breaking' 'safest' 'half' 'repair' 'various' 'modern' 'repaired'
 'updated' 'sophisticated']
```

For the top words, we can identify descriptive words that people associate with the brand.

Audi is more associated with words such as turbo and supercharged.

Lexus is more associated with words such as quiet, smooth, and luxurious.

INFINTI is more associated with words such as intelligent, spacious, and dependable.

BMW is more associated with words such as ultimate and convertible.

Mercedes-Benz is more associated with words such as safest, modern, and sophisticated.

## Exercises

1. List at least three different roles and responsibilities of marketing departments.
2. List the differences between sales teams and marketing teams for each of the following points:
   - Objectives
   - Methods
   - Scope
   - Responsibilities
   - Tools used
3. Identify three different challenges that human resource departments face.
4. For each of the challenges listed in 3, identify one way that artificial intelligence could potentially be used to alleviate the problem.
5. Predict a marketing lead score for potential term deposit customers for a bank.
   We will utilize this dataset [1]:
   https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset
   This dataset consists of information collected about various leads that were previously contacted and whether or not they eventually opened up a deposit account.
   In this exercise, try to use what you have learned so far to develop a model that can provide a marketing lead score to help marketers identify suitable potential customers for targeted advertising.
6. Perform customer segmentation on automobile customers.
   We will utilize this dataset [2]:
   https://www.kaggle.com/datasets/akashdeepkuila/automobile-customer

This dataset consists of information collected about various automobile customers.

In this exercise, discard the segmentation grouping provided and try to use what you have learned so far to develop a model that can segment the customers into different groups for better targeted marketing.

## References

1. Bachmann JM (2017) Bank marketing dataset. https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset
2. Kuila A (2021) Automobile customer. https://www.kaggle.com/datasets/akashdeepkuila/automobile-customer
3. Vjchoudhary7 (2018) Mall customer segmentation data. In: Kaggle. https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python. Accessed 27 Apr 2023

# Chapter 12
# AI in Supply Chain Management


**Learning Outcomes**

- Understand the roles and responsibilities of supply chain departments.
- Be able to explain the various segments of the supply chain cycle.
- Be able to list examples of supply chain models.
- Be able to define what is the bullwhip effect.
- Be able to list potential causes of variation in quantity ordered.
- Recommend ways to reduce the bullwhip effect.
- Identify ways that artificial intelligence can be applied to supply chain management.
- Develop artificial intelligence solutions for demand forecasting, automating quality assurance, predicting delivery time, and optimizing deliveries.

## 12.1 Introduction to Supply Chain Management

### *12.1.1 Supply Chain Definition*

A supply chain is an interrelated series of processes within a firm and across different firms covering the entire chain of supply and operations from raw materials to the end customer [4].

Supply chain management is a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that merchandise is produced and distributed in the right quantities, and at the right time, in order to minimize system-wide costs while satisfying service level requirements.

By optimizing the supply chain, businesses can reduce extra expenses and expedite product delivery to consumers. This is accomplished by maintaining tighter control over internal inventories, internal manufacturing, distribution channels, sales, and vendor stocks.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_12

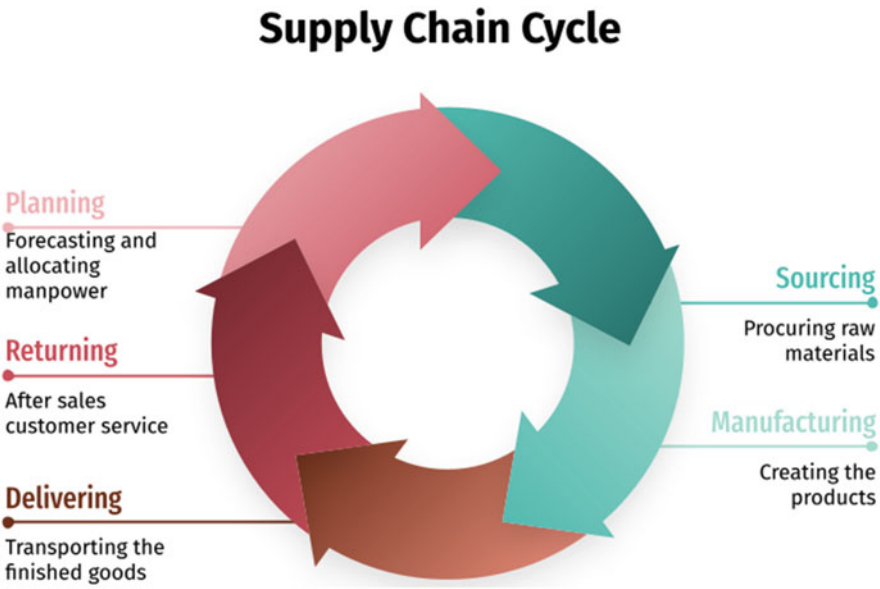



**Fig. 12.1** The supply chain cycle

Supply chain cycle as shown in Fig. 12.1 consists of 5 operations which are planning, sourcing, manufacturing, delivering, and returning.

1. **Planning**
   The supply chain management (SCM) process often begins with planning to align supply with customer and production expectations. Companies are required to forecast their future demands for the availability and quantity of raw materials required at various points in the SCM process, as well as the capabilities and constraints of relevant equipment and the number of workers required. Large organizations frequently rely on Enterprise Resource Planning (ERP) system modules to aggregate data and develop these plans.
2. **Sourcing**
   Effective SCM operations rely primarily on reliable supplier relationships. Sourcing requires collaborating with suppliers to obtain the necessary raw materials for the production process. A corporation may be able to plan ahead and collaborate with a supplier to procure goods. However, several businesses have distinct sourcing requirements. Typically, SCM sourcing involves ensuring:

   - The raw materials satisfy the manufacturing requirements necessary for the manufacture of commodities.
   - The prices paid are in accordance with market expectations.
   - Due to unforeseen situations, the provider has the ability to send emergency supplies.
   - The supplier has a track record of on-time high-quality product delivery.

When businesses are working with perishable commodities, supply chain management is very crucial. When sourcing materials, businesses should consider lead time and a supplier's ability to meet those requirements.

3. **Manufacturing**
   Supply chain management revolves around the company's ability to create something new out of raw materials through the application of energy, time, and other resources. This final product is the end result of the manufacturing process, although it is not the last step in supply chain management.
   The manufacturing procedure can be subdivided into tasks such as assembly, testing, inspection, and packing. During the production process, a company must be cautious of waste and other controllable elements that may result in deviations from the plans. For instance, if a company uses more raw materials than intended and sourced for due to a lack of employee training, the organization must address the issue or revisit the previous stages of SCM.
4. **Delivering**
   Once things are produced and sales are completed, a business must deliver them to customers. Distribution is frequently viewed as an important factor to the organization's brand image since the customer has not yet interacted with the product at this point. An organization with excellent SCM procedures has reliable logistical resources and distribution channels to guarantee the timely, risk-free, and cost-effective transport of goods. This includes having a backup distribution method or multiple distribution methods in the event that one source of transportation is temporarily unavailable.
5. **Returning**
   The management of the supply chain finishes with support for product and customer returns. It is bad when a client must return a product and is worse when the return is due to an error on the company's side. This procedure is referred to as reverse logistics, and the organization must have the capacity to handle returned products and appropriately issue refunds. Regardless of the reason for a return, be it a product recall or customer dissatisfaction, the return has to executed and the underlying issue has to be rectified.
   Many view customer returns as a form of contact between the customer and the business. However, the communication between businesses to determine the cause of a defect is crucial to the success of the customer return process. Without addressing the root cause of a customer return, subsequent returns are likely to continue.

### *12.1.2 Types of Supply Chain Models*

Supply chain management differs from company to company. Every organization's SCM process is shaped by its own set of objectives, limitations, and advantages. Typically, there are six primary models a firm can utilize to direct its supply chain management activities.

- **Continuous Flow Model**
  This model, which is one of the more classic supply chain strategies, is frequently best suited for mature industries. The continuous flow approach relies on a company manufacturing the same product again and anticipating minimal volatility in consumer demand.
- **Agile Model**
  This model is ideal for businesses whose demand is unpredictable or whose products are made to order. This approach promotes adaptability, as a company may at any time have a special need and must be prepared to pivot accordingly. This model is usually suited for a product with a short life cycle to have rapid turnovers.
- **Fast Model**
  A business seeks to capitalize on a trend, produce goods rapidly, and ensure that all products are sold before the trend ends when using a fast chain model.
- **Flexible Model**
  The flexible model is most effective for seasonal businesses. Some businesses may have substantially higher demand requirements during peak season, while others may have minimal volume requirements. A flexible form of supply chain management facilitates the ramping up or winding down of production.
- **Efficient Model**
  For businesses operating in industries with extremely slim profit margins, the most efficient supply chain management technique may provide a competitive advantage. This comprises the optimal utilization of equipment and machinery, as well as the management of inventories and processing of orders.
- **Custom Model**
  If none of the aforementioned models meet a company's demands, it can always choose for a custom model. This is typically the case in highly specialized businesses with stringent technological standards, such as the automobile industry.

### *12.1.3 Bullwhip Effect*

The various parties in the supply chain such as customers, suppliers, manufacturers, and merchants have minimal power over the supply chain, yet their activities have an impact on everyone else [1]. As these parties attempt to respond to demand changes, ripple effects propagate down the supply chain. This effect is known as the bullwhip effect. This phenomenon was named after the movement of a bullwhip, in which a slight wrist movement results in a considerably bigger, uncontrolled movement at the whip's end.

The bullwhip effect occurs in a supply chain when each party steadily escalates an originally tiny increase in demand, causing the demand to spiral out of control. Each participant in the supply chain overcompensates for this demand with excess demand for raw materials, which results in higher output, erroneous demand forecasting, and uneven inventory.

### 12.1.4 Causes of Variation in Orders

- **Structural**
  - **Order batching**: To keep things simple, members of the supply chain may round their orders up or down, or they may wait until a given date to place an order which makes forecasting demand challenging.
  - **Promotion**: Discounts, promotions, and other special offers have an impact on ordinary demand and can lead to erroneous estimates as buyers try to adjust their usual forecasting metrics for these deals.
  - **Long lead time**: If replenishment merchandise takes a long time to reach the seller, it will not be in sync with demand and will hinder the seller's ability to meet client demands.
- **Behavioural**
  - **Shortage gaming**: When upstream inventory becomes low, retailers and suppliers order substantially higher quantities and build up their own supplies to ensure they can satisfy demand, often harming the entire supply chain in the process.
  - **Over reaction to backlog**: Meeting client demands frequently necessitates extra options, such as in-store pickups and direct-from-vendor shipments. The supply chain may become more complex as a result of these various requirements, which may also raise the pressure to have supplies on hand. Overstocking may result from trying to ensure a supply of all possible choices.
  - **Fear of empty stock**: Members of the supply chain may fear having an empty stock and losing customers as a result. Under high demand, they may place excessive orders and cause a bullwhip effect.

### 12.1.5 Reducing the Bullwhip Effect

The bullwhip effect is exacerbated because supply chain stakeholders lack a comprehensive understanding of why buyers are raising demand. Improving chain-wide visibility can help everyone understand the context of demand shifts. For example, the rise in orders could be attributable to a promotion, seasonal demands, or something else. Members can identify potential causes of overreactions and resolve them before the situation spirals out of control.

Having precise, current demand information is the best method to combat the bullwhip effect [8]. To do this, we must switch from a forecast-driven ordering system to the one that enables information sharing with supply chain partners and, hence, full awareness of actual customer demand. Kanban System, Vendor Managed Inventory (VMI), Strategic Supply Chain Partnerships, Lean Management, Real-Time Information Sharing, and Just-in-Time Inventory Replenishment System are a

few of the common measures that companies around the world employ to combat the bullwhip effect and establish a demand-driven supply chain.

- **Vendor Managed Inventory**
  VMI programs enable suppliers to obtain sales and forecast data from downstream partners in real time. In conjunction with predetermined settings such as minimum/maximum shelf presence, the data is analyzed by an algorithm for machine learning to generate replenishment recommendations. By utilizing VMI, suppliers no longer have to wait for retailers or distributors to run out of stock before replenishing. Instead, businesses can plan for a new shipment to arrive at the optimal time to avoid stockouts and excess inventory.
- **Kanban System**
  Kanban is a tool for optimizing the supply chain that organizations can employ to increase productivity. It is a lean or just-in-time manufacturing method in which just the necessary components and stock are replenished at that time. This allows businesses to increase productivity while decreasing inventories.
- **Strategic Supply Chain Partnerships**
  Reducing the product's required travel distance by sourcing for local suppliers can reduce lead times and expenses. This approach is not always realistic, but it is always possible to locate dependable, swift partners that can help to reduce lead times. Strategic collaboration involves working with your partners to increase forecast accuracy, fortify connections, and head off problems before they even happen. Aligning your key performance indicators (KPIs) and other performance measurements can assist in keeping everyone on the same page. Strong collaboration is one of the most effective safeguards against the bullwhip effect, which is typically caused by fragmented inventory practices.
- **Lean Management**
  Lean supply chain management entails minimizing waste and saving costs as a result. The lean strategy emphasizes efficient, simplified processes, and the elimination of anything that does not provide value to the customer-facing product or service. Lean supply chain management prioritizes dependability and predictability over adaptability and flexibility.
- **Real-Time Information Sharing**
  To lessen the impact of the "bullwhip effect," participants in the supply chain should coordinate their actions. Sharing information is very important in this case because it lets different groups work together and see more of the supply chain than just the part they control. Collaboration is of utmost importance in increasingly globalized supply chains, as items may cross international borders and pass through numerous businesses.
- **Just-in-Time Inventory Replenishment System**
  Just-in-time inventory management ensures that inventory arrives precisely when it is required for manufacturing or to fulfill consumer demand and no earlier. The objective is to eliminate waste and improve operation efficiency. Long-term contracts with dependable suppliers are required for this to work because the primary focus is on frequency quality and not pricing. Just-in-time is an example

of lean management. All components of a manufacturing or service system, including people, are integrated in just-in-time. They are interdependent and mutually dependent on producing good outcomes.

By incorporating one or more of the aforementioned methods and management philosophies, we can not only mitigate the bullwhip effect in our supply chain but also benefit from improved long-term strategic alliances.

## 12.2 Artificial Intelligence in Supply Chain Management

Artificial intelligence has many use cases in supply chain management and can bring about advancements in many segments of the supply chain [7]. These applications include:

1. **Back-office automation**
   By combining conversational AI with Robotic Process Automation (RPA), labor-intensive and repetitive tasks such as document processing can be automated.
2. **Logistics automation**
   Artificial intelligence and automated processes can also help improve supply chain logistics. Major investments in transport automation technology, such as driverless trucks, would be able to improve the efficiency and scale of our supply chain.
3. **Warehouse automation**
   Automatic warehouse management is bolstered by AI-enabled technologies such as collaborative robots which improves efficiency, productivity, and safety of warehouses.
4. **Automated quality checks**
   Product quality checks can be aided through the use of AI-powered computer vision systems. Since these systems do not become exhausted, they can improve productivity and accuracy in the quality assurance process. Through employing computer vision, defects can be detected automatically, thus automating and improving the quality of products.
5. **Automated inventory management**
   Bots with computer vision and AI/ML capabilities can be used to automate repetitive operations in inventory management, such as real-time inventory scanning. Inventory scanning bots like this can also be used in retail businesses. However, before implementing such solutions, it is important to analyze their feasibility and assess their long-term benefits.
6. **Inventory optimization**
   By examining historical demand and supply data and patterns, AI-powered systems can assist in determining ideal inventory levels.

7. **Region-specific forecasts**
   The supply chain AI can also provide comprehensive regional demand data to assist corporate leaders in making better decisions. For example, each region has its own festivals, holidays, fashions, and so on. AI-powered forecasting systems can help customize fulfillment procedures based on region-specific requirements by employing region-specific parameters.
8. **Bullwhip effect prevention**
   The bullwhip effect is a significant source of concern in supply chain management. This is because tiny changes at one end of the supply chain are amplified as they move upstream/downstream. By integrating data collected from customers, suppliers, manufacturers, and distributors, AI-powered forecasting solutions can help mitigate demand and supply swings to regulate the bullwhip effect. This would lead to the reduction of stockouts and backlogs.
9. **Improved supplier selection**
   AI-enabled supplier relationship management (SRM) software can help with supplier selection based on aspects like pricing, buying history, sustainability, and so on. AI-powered systems can also assist in tracking, analyzing, and ranking supplier performance data.
10. **Improved supplier communications**
    RPA and other AI-powered tools can also help automate typical supplier communications such as invoice sharing and payment reminders. Automation of these activities can aid in avoiding delays, such as forgetting to pay a vendor on time, which can have a detrimental impact on shipment and output.
11. **Greener transport logistics**
    Using factors like traffic, road closures, and weather, AI-powered systems can help optimize transportation routes by reducing the distance traveled. AI can be used, for instance, to optimize vehicle routes and reduce fuel use, resulting in reduced emissions and enhanced sustainability.
12. **Greener warehousing**
    Carbon emissions associated with excess inventory storage and transit can be minimized through employing AI in maintaining ideal inventory levels. Smart energy usage solutions can also help to reduce greenhouse gas emissions associated with warehouse energy consumption.
13. **Route Optimization**
    Shorter and more cost-efficient routes can be found and planned using AI to help optimize and reduce the costs incurred by suggesting better routes. Real-time route suggestion can also be done to help reduce implications of road conditions such as weather and traffic jams.
14. **Resource Allocation**
    Artificial intelligence can also help to automate resource planning requirements through taking into account the production and requirements of different segments in the supply chain such that the number of deliveries and time taken to transport logistics are minimized.

## 12.3 Applications of AI in Supply Chain Management

### *12.3.1 Demand Forecasting with Anomaly Detection*

AI in demand forecasting can help to predict micro- and macro-trends used for optimizing the amount of goods to be produced and help determine the amount of raw materials to be purchased from suppliers. This will help to reduce the bullwhip effect brought about by over or under productions through generating better consumer demand and supply estimates.

Anomalous demand can also be flagged out by the system when the actual demand is significantly more or less than forecasted. This would enable managers to determine whether the effect is due to a long-term trend or a short-term spike to guide future predictions.

Through the use of AI in demand forecasting, organizations would be able to capture macro- and micro-trends in consumer behaviors quickly and accurately, as well as reduce the time needed to react to drivers of change in demand, thus bringing improved profitability to the organizations.

In this example we will use the pytrends library to extract the worldwide Google trends for sunscreen between 1 Jan 2017 to 1 Jan 2022. This will give us an estimate of the interest in a particular type of product.

```
import pytrends
from pytrends.request import TrendReq
import pandas as pd
```

```
tr = TrendReq(hl='en-US', tz=360)
tr.build_payload(kw_list = ["sunscreen"], cat=0,␣
↪timeframe='2017-01-01 2022-01-01', geo='', gprop='')
```

tr.interest_over_time() returns a pandas dataframe object. We will reset the index such that the index will correspond to the row number. We will print the top few rows using head to visualize what the dataframe looks like.

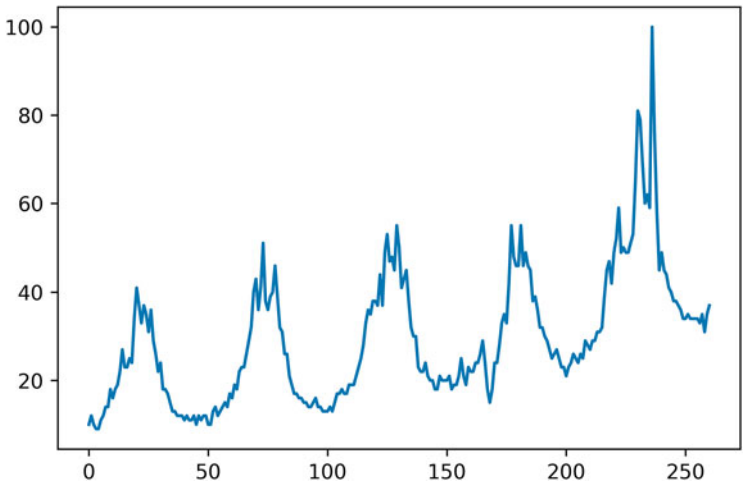
```
df = tr.interest_over_time().reset_index()
df.head()
```

```
        date  sunscreen  isPartial
0 2017-01-01         11      False
1 2017-01-08         12      False
2 2017-01-15         11      False
3 2017-01-22         10      False
4 2017-01-29         11      False
```

```
# df = pd.read_csv("Sunscreen_Google_Trends.csv")
```

As you can see, the data contains the relative number of searches for sunscreen, where 100 is the date with the highest search and 0 is the date with the least amount of searches. We can plot this to visualize what the demand looks like over time.

```
plt.plot(df["sunscreen"])
plt.show()
```




From the graph, we can see that sunscreen searches are increasing over time and exhibit some form of annual seasonality, likely due to higher demands in sunscreen during summer months where the sun is the hottest within a year.

We will continue by splitting our data into training and testing, where the testing data is the last 20% of dates we extracted.

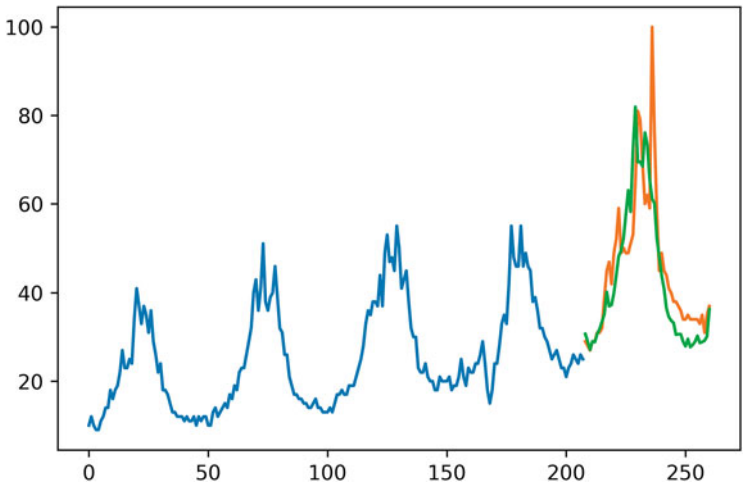
```
from statsmodels.tsa.api import SimpleExpSmoothing,␣
↪Holt, ExponentialSmoothing
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np

n_row=len(df)
train_row = int(0.8 * n_row)
train = df[0:train_row]
test = df[train_row:]
```

Let us train our model with an additive trend and multiplicative seasonality as we observed from the data collected.

```
model = ExponentialSmoothing(train["sunscreen"], trend='add', seasonal='mul', seasonal_periods=52).fit()
pred = model.forecast(len(test))
```

```
# model = ExponentialSmoothing(train["sunscreen"], trend='add').fit()
# pred = test.copy()
# pred['Holt_Winter'] = model.forecast(len(test))
# pred = pred.dropna()
print(mean_squared_error(pred, test["sunscreen"])**0.5)

plt.plot(train["sunscreen"])
plt.plot(test["sunscreen"])
plt.plot(pred)
plt.show()
```

```
10.310581438285055
```



From the graph, it seems that our model performs relatively well. We can use this model to forecast the increase in demand for sunscreen over time. However, there are certain occasions in the year where the demand is significantly higher than our forecast.

Such instances can be flagged out through anomaly detection methods applied on our model, so that the organization can assess whether the change in trend is something to be concerned about.

In this example, we will use the Brutlag algorithm for anomaly detection to add on the Holt winters time series [2].

Confidence bands are bands which classify whether a data point is considered an anomaly. Data points are considered as anomalies when they go above the upper band or below the lower band. The confidence bands for the Brutlag algorithm are as follows [2]:

$$\hat{y}_{max_t} = L_{t-1} + P_{t-1} + S_{t-p} + md_{t-p}$$
$$\hat{y}_{min_t} = L_{t-1} + P_{t-1} + S_{t-p} - md_{t-p}$$
$$d_t = \gamma|y_t - \hat{y}_t| + (1-\gamma)d_{t-p}$$
$$\hat{y}_{max_t} = \hat{y}_t + m(\gamma|y_{t-p} - \hat{y}_{t-p}| + (1-\gamma)d_{t-2p})$$
$$\hat{y}_{min_t} = \hat{y}_t - m(\gamma|y_{t-p} - \hat{y}_{t-p}| + (1-\gamma)d_{t-2p})$$

where

$k$ represents number of measurements in time series.
t represents the moment in time.
$y_t$ represents the predicted value of variable at moment $t$.
$y_t$ represents the real measured observation in moment $t$.
$p$ represents the time series period.
α represents the data smoothing factor.
β represents the trend smoothing factor.
γ represents the seasonal change smoothing factor
$m$ represents the scaling factor for Brutlag confidence band.

Let us begin by calculating the difference between the data points and our model's prediction.

```
predictions = model.predict(start=df.iloc[:, 1].
 ↪index[0], end=df.iloc[:, 1].index[-1])
diff = df["sunscreen"] - predictions
```

```
"""Brutlag Algorithm"""
PERIOD = 52                                                    # The␣
↪given time series has seasonal_period=52
GAMMA = 0.10                                                   # the␣
↪seasonility component
```

(continues on next page)

(continued from previous page)

```
SF = 2.75                                                                #␣
↪brutlag scaling factor for the confidence bands.
UB = []                                                                  #␣
↪upper bound or upper confidence band
LB = []                                                                  #␣
↪lower bound or lower confidence band
```

```
difference_array = []
dt = []
difference_table = {
    "actual": df["sunscreen"], "predicted":␣
↪predictions, "difference": difference_array, "UB":␣
↪UB, "LB": LB}
```

Let us calculate the upper and lower confidence bands.

```
for i in range(len(predictions)):
    diff = df["sunscreen"][i]-predictions[i]
    if i < PERIOD:
        dt.append(abs(diff/predictions[i]))
    else:
        dt.append(GAMMA*abs(diff/predictions[i]) + (1-
↪GAMMA)*dt[i-PERIOD])

    difference_array.append(diff)
    UB.append(predictions[i] +␣
↪abs(predictions[i])*SF*dt[i])
    LB.append(predictions[i] -␣
↪abs(predictions[i])*SF*dt[i])
```

```
print("\nDifference between actual and predicted\n")
difference = pd.DataFrame(difference_table)
print(difference)
```

```
Difference between actual and predicted

     actual  predicted  difference         UB         LB
0        11  11.262954   -0.262954  11.986079  10.539830
1        12  10.792813    1.207187  14.112578   7.473047
2        11  11.041324   -0.041324  11.154965  10.927683
3        10  11.857316   -1.857316  16.964936   6.749697
4        11  10.188302    0.811698  12.420471   7.956134
..      ...        ...         ...        ...        ...
256      35  27.752998    7.247002  35.277667  20.228328
257      32  28.794269    3.205731  31.104807  26.483730
258      30  28.718575    1.281425  33.757684  23.679466
```

(continues on next page)

(continued from previous page)

```
259      34  28.650632    5.349368  32.877352  24.423912
260      37  34.076066    2.923934  37.206145  30.945987

[261 rows x 5 columns]
```

Now that we have determined the upper and lower confidence bands, we can classify each data point as either normal or an anomaly.

```
normal = []
normal_date = []
anomaly = []
anomaly_date = []
```

```
for i in range(len(df["sunscreen"].index)):
    if (UB[i] < df["sunscreen"][i] or LB[i] > df[
↪"sunscreen"][i]) and i > PERIOD:
        anomaly_date.append(df.index[i])
        anomaly.append(df["sunscreen"][i])
    else:
        normal_date.append(df.index[i])
        normal.append(df["sunscreen"][i])
```

```
anomaly = pd.DataFrame({"date": anomaly_date, "value
↪": anomaly})
anomaly.set_index('date', inplace=True)
normal = pd.DataFrame({"date": normal_date, "value":␣
↪normal})
normal.set_index('date', inplace=True)
```

```
print(len(anomaly), "data points classified as␣
↪anomaly\n")
```

```
40 data points classified as anomaly
```

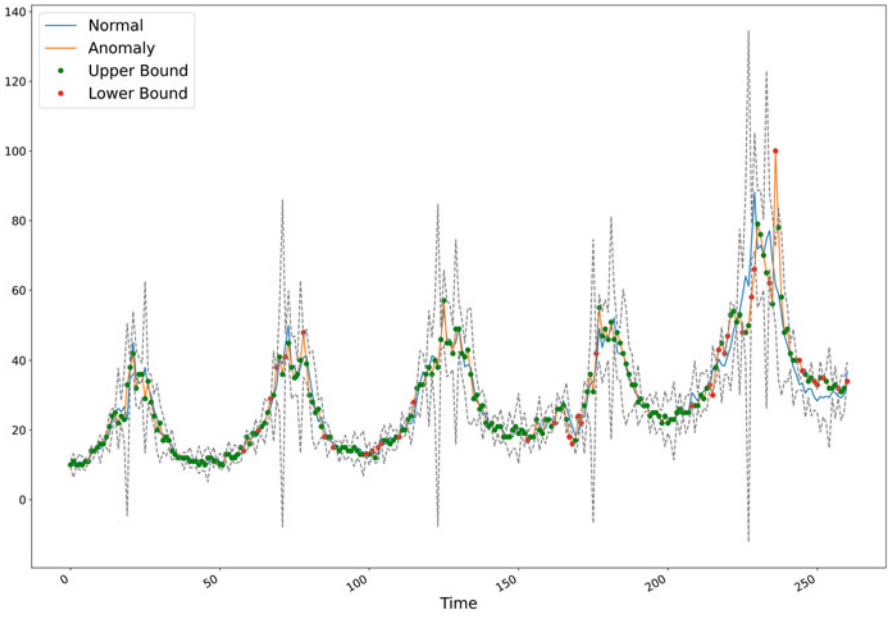
Now let us visualize our work so far.

```
"""
Plotting the data points after classification as␣
↪anomaly/normal.
Data points classified as anomaly are represented in␣
↪red and normal in green.
"""
plt.figure(figsize=(20,20))
```

(continues on next page)

(continued from previous page)

```
plt.plot(predictions.index, predictions)
plt.plot(df.index, df["sunscreen"])
plt.plot(normal.index, normal, 'o', color='green')
plt.plot(anomaly.index, anomaly, 'o', color='red')
# Ploting brutlag confidence bands
plt.plot(df.index, UB, linestyle='--', color='grey')
plt.plot(df.index, LB, linestyle='--', color='grey')
# Formatting the graph
plt.legend(['Normal', 'Anomaly', 'Upper Bound',
 ↪'Lower Bound'])
plt.gcf().autofmt_xdate()
plt.xlabel("Time")
plt.show()
```




### 12.3.2 *Quality Assurance*

One of the other significant use cases of applying artificial intelligence to the supply chain is in quality assurance. Through the use of computer vision, quality assurance in the supply chain can be automated to work round the clock with good accuracy rates. This can help to reduce the number of defective products shipped out, therefore increasing customer satisfaction and strengthen brand reliability.

For this exercise, we will be using the dataset from [5]:
https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting-product

The dataset will be used to detect defects in a casting manufacturing product. Casting is a manufacturing process in which a liquid material is usually poured into a mold, which contains a hollow cavity of the desired shape, and then allowed to solidify. However, casting defects may occur in these casts which are undesired irregularities formed during the metal casting process and need to be detected.

There are many types of defects in casts such as blow holes, pinholes, burr, shrinkage defects, mold material defects, pouring metal defects, metallurgical defects, etc.

The quality inspection department is in charge of ensuring the quality of the casts. However, the inspection process is carried out manually and is a very time-consuming process which is subjected to human error and is not 100% accurate as well. Thus we will create a model to classify between acceptable casts and defective casts automatically.

These all photos are top view of a submersible pump impeller. The dataset contains the total 7348 image data in total. These all are the size of (300*300) pixels gray-scaled images. In all images, data augmentation has already been applied.

There are mainly two categories:

1. Defective
2. Ok

The data have been split with the following distribution:
train: deffront has 3758 images and okfront has 2875 images
test: deffront has 453 images and ok_front has 262 images

In this example, we will apply transfer learning by using the weights of some other model previously trained on a separate task. We will achieve this by training only the last few layers of our model. Transfer learning is effective in training models with small datasets as it uses pre-trained features from other big datasets and simply finetunes the network for the new task. This works because the features learned by CNNs on larger datasets are able to capture a variety of descriptive features that are sufficient to describe the data, and thus we can simply train the last layer to classify between the defective samples and the normal samples.

We will begin by importing the relevant packages and visualizing the datasets used for training.

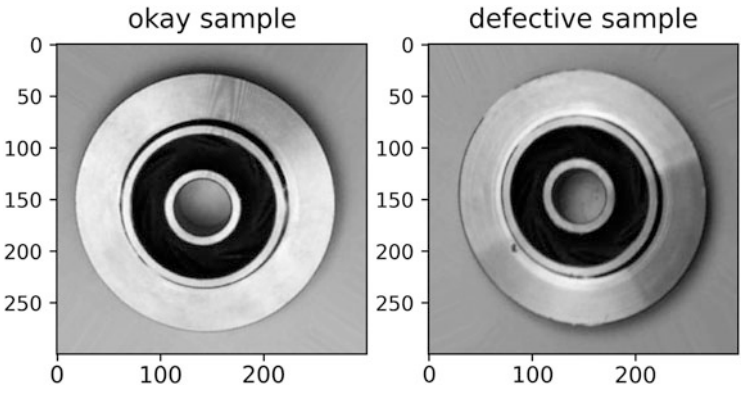
```
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt

import tensorflow, os
from tensorflow.keras.layers import Conv2D, MaxPool2D,
↪ Dropout, Flatten, Dense
```

(continues on next page)

(continued from previous page)

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
```

```
okay_folder = "Supply_Chain/casting_data/casting_data/
↪train/ok_front"
okay_sample = os.path.join(okay_folder, os.
↪listdir(okay_folder)[0])
okay_image = cv2.cvtColor(cv2.imread(okay_sample),␣
↪cv2.COLOR_BGR2RGB)
fig, axs = plt.subplots(1, 2)
axs[0].imshow(okay_image)
axs[0].set_title("okay sample")

def_folder = "Supply_Chain/casting_data/casting_data/
↪train/def_front"
def_sample = os.path.join(def_folder, os.listdir(def_
↪folder)[0])
def_image = cv2.cvtColor(cv2.imread(def_sample), cv2.
↪COLOR_BGR2RGB)
axs[1].imshow(def_image)
axs[1].set_title("defective sample")
plt.show()
```




As you can see, there are tiny dents in the defective sample that differentiates it from the okay samples. We will train our convolutional neural network to differentiate between the two. To do that, we will first have to create our data generators.

```
train_data_gen = image.ImageDataGenerator(rescale= 1./255)
train= train_data_gen.flow_from_directory(directory="Supply_Chain/casting_
↪data/casting_data/train" , target_size=(256,256) , batch_size=32, class_
↪mode = 'binary')
```

```
Found 6633 images belonging to 2 classes.
```

```
train_data_gen = image.ImageDataGenerator(rescale= 1./
↪255)
test= train_data_gen.flow_from_directory(directory=
↪"Supply_Chain/casting_data/casting_data/test" ,␣
↪target_size=(256,256) , batch_size=32, class_mode =
↪'binary')
```

```
Found 715 images belonging to 2 classes.
```

In this case we will be finetuning the Xception network trained original on ImageNet.

We need to specify the input shapes for the images we will feed the model and set include_top to False as we want to include other layers to the end the Xception model to be trained on our own data.

```
from tensorflow.keras.applications import Xception
xcept = Xception(input_shape = (256, 256, 3), include_
↪top = False, weights = 'imagenet')
```

In this case we will set the xcept.layers to not be trainable as we want to preserve the already trained weights. We will just add additional layers to the end for the classification task.

```
for layer in xcept.layers:
    layer.trainable = False


model = tensorflow.keras.Sequential([
    xcept,
    tensorflow.keras.layers.Flatten(),
    tensorflow.keras.layers.Dense(units=256,␣
↪activation="relu"),
    tensorflow.keras.layers.Dropout(0.2),
    tensorflow.keras.layers.Dense(units=1, activation=
↪"sigmoid"),
])
```

(continues on next page)

(continued from previous page)

```
model.compile(optimizer="adam",loss='binary_
↪crossentropy',metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 8, 8, 2048)        20861480
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 256)               33554688
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 54,416,425
Trainable params: 33,554,945
Non-trainable params: 20,861,480
_________________________________________________________________
```

```
model.fit(train,epochs=10,steps_per_epoch=7,
↪validation_data=test,validation_steps=len(test))
```

```
Epoch 1/10
7/7 [==============================] - 21s 3s/step - loss: 12.7714 - accuracy: 0.
↪5848 - val_loss: 2.0342 - val_accuracy: 0.7105
Epoch 2/10
7/7 [==============================] - 19s 3s/step - loss: 1.9719 - accuracy: 0.
↪7455 - val_loss: 0.9434 - val_accuracy: 0.8322
Epoch 3/10
7/7 [==============================] - 19s 3s/step - loss: 1.0524 - accuracy: 0.
↪8616 - val_loss: 1.1076 - val_accuracy: 0.8126
Epoch 4/10
7/7 [==============================] - 19s 3s/step - loss: 0.9593 - accuracy: 0.
↪8616 - val_loss: 0.7977 - val_accuracy: 0.8629
Epoch 5/10
7/7 [==============================] - 19s 3s/step - loss: 1.0131 - accuracy: 0.
↪8661 - val_loss: 0.4438 - val_accuracy: 0.9119
Epoch 6/10
7/7 [==============================] - 19s 3s/step - loss: 0.6367 - accuracy: 0.
↪8973 - val_loss: 0.3567 - val_accuracy: 0.9287
Epoch 7/10
7/7 [==============================] - 19s 3s/step - loss: 0.7191 - accuracy: 0.
↪8705 - val_loss: 0.4454 - val_accuracy: 0.9105
Epoch 8/10
7/7 [==============================] - 19s 3s/step - loss: 0.3637 - accuracy: 0.
↪9554 - val_loss: 0.1467 - val_accuracy: 0.9706
Epoch 9/10
7/7 [==============================] - 19s 3s/step - loss: 0.0665 - accuracy: 0.
↪9777 - val_loss: 0.1938 - val_accuracy: 0.9566
```

(continues on next page)

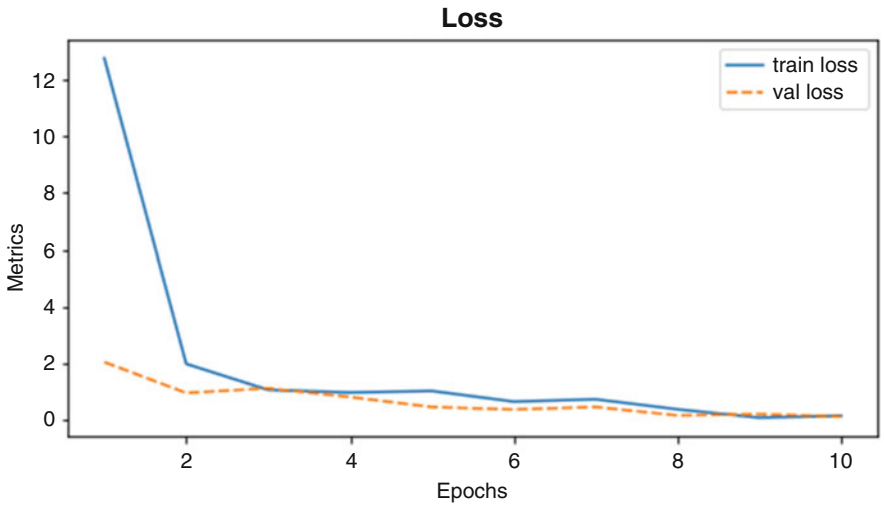
(continued from previous page)

```
Epoch 10/10
7/7 [==============================] - 19s 3s/step - loss: 0.1318 - accuracy: 0.
↪9732 - val_loss: 0.1082 - val_accuracy: 0.9776
```

```
<tensorflow.python.keras.callbacks.History at 0x1fc28d5fa00>
```

As you can see, the model is able to predict the dataset well with up to 97.76% accuracy. Let us plot the loss and accuracy graphs.

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.subplots(figsize = (8, 4))
df = pd.DataFrame(model.history.history, index =␣
↪range(1, 1+len(model.history.epoch)))
sns.lineplot(data = df[["loss", "val_loss"]])
plt.title("Loss", fontweight = "bold", fontsize = 20)
plt.xlabel("Epochs")
plt.ylabel("Metrics")

plt.legend(labels = ['train loss', 'val loss'])
plt.show()
```

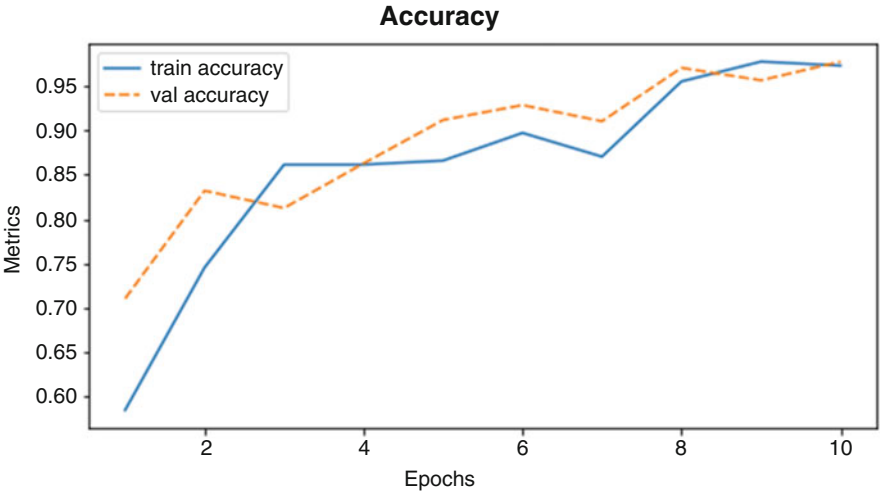



```
plt.subplots(figsize = (8, 4))
sns.lineplot(data = df[["accuracy", "val_accuracy"]])
plt.title("Accuracy", fontweight = "bold", fontsize =␣
↪20)
```

(continues on next page)

(continued from previous page)

```
plt.xlabel("Epochs")
plt.ylabel("Metrics")

plt.legend(labels = ['train accuracy', 'val accuracy
↪'])
plt.show()
```




### *12.3.3 Estimating Delivery Time*

In order to allocate resources efficiently, supply chain managers have to be able to estimate the time required to transfer one resource or product from one place to another. Underestimating the time required to deliver a resource may lead to delays down the supply chain and overestimating the time required would lead to inefficient resource allocation. Thus it is important for supply chain managers to be able to monitor and predict how long their deliveries would take. Artificial intelligence is able to take into account information from a multitude of sources in order to accurately predict the delivery time, so that supply chain managers can plan accordingly. In this example, we will develop a model to predict the delivery time for food deliveries from the restaurant to the delivery location.

The dataset that we will be using will be taken from this link [3]:

https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=train.csv

Let us begin by importing the packages required and the data.

```
import pandas as pd
import numpy as np
from geopy.distance import geodesic
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score, mean_absolute_
↪error
```

```
df = pd.read_csv("Supply_Chain/train.csv")
df
```

```
            ID Delivery_person_ID Delivery_person_Age Delivery_person_Ratings  \
0      0x4607     INDORES13DEL02                  37                     4.9
1      0xb379     BANGRES18DEL02                  34                     4.5
2      0x5d6d     BANGRES19DEL01                  23                     4.4
3      0x7a6a    COIMBRES13DEL02                  38                     4.7
4      0x70a2     CHENRES12DEL01                  32                     4.6
...       ...                ...                 ...                     ...
45588  0x7c09      JAPRES04DEL01                  30                     4.8
45589  0xd641      AGRRES16DEL01                  21                     4.6
45590  0x4f8d     CHENRES08DEL03                  30                     4.9
45591  0x5eee    COIMBRES11DEL01                  20                     4.7
45592  0x5fb2   RANCHIRES09DEL02                  23                     4.9

       Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0                22.745049             75.892471                   22.765049
1                12.913041             77.683237                   13.043041
2                12.914264             77.678400                   12.924264
3                11.003669             76.976494                   11.053669
4                12.972793             80.249982                   13.012793
...                    ...                   ...                         ...
45588            26.902328             75.794257                   26.912328
45589             0.000000              0.000000                    0.070000
45590            13.022394             80.242439                   13.052394
45591            11.001753             76.986241                   11.041753
45592            23.351058             85.325731                   23.431058

       Delivery_location_longitude  Order_Date Time_Orderd Time_Order_picked  \
0                        75.912471  19-03-2022    11:30:00          11:45:00
1                        77.813237  25-03-2022    19:45:00          19:50:00
2                        77.688400  19-03-2022    08:30:00          08:45:00
3                        77.026494  05-04-2022    18:00:00          18:10:00
4                        80.289982  26-03-2022    13:30:00          13:45:00
...                            ...         ...         ...               ...
45588                    75.804257  24-03-2022    11:35:00          11:45:00
45589                     0.070000  16-02-2022    19:55:00          20:10:00
45590                    80.272439  11-03-2022    23:50:00          00:05:00
45591                    77.026241  07-03-2022    13:35:00          13:40:00
45592                    85.405731  02-03-2022    17:10:00          17:15:00

          Weatherconditions Road_traffic_density  Vehicle_condition  \
```

(continues on next page)

(continued from previous page)

```
0           conditions Sunny                High                   2
1          conditions Stormy                 Jam                   2
2      conditions Sandstorms                 Low                   0
3           conditions Sunny              Medium                   0
4          conditions Cloudy                High                   1
...                      ...                 ...                 ...
45588       conditions Windy                High                   1
45589       conditions Windy                 Jam                   0
45590      conditions Cloudy                 Low                   1
45591      conditions Cloudy                High                   0
45592         conditions Fog              Medium                   2

      Type_of_order Type_of_vehicle multiple_deliveries Festival  \
0             Snack      motorcycle                   0      No
1             Snack         scooter                   1      No
2            Drinks      motorcycle                   1      No
3            Buffet      motorcycle                   1      No
4             Snack         scooter                   1      No
...             ...             ...                 ...     ...
45588         Meal      motorcycle                   0      No
45589       Buffet      motorcycle                   1      No
45590       Drinks         scooter                   0      No
45591        Snack      motorcycle                   1      No
45592        Snack         scooter                   1      No

                City Time_taken(min)
0              Urban          (min) 24
1      Metropolitian          (min) 33
2              Urban          (min) 26
3      Metropolitian          (min) 21
4      Metropolitian          (min) 30
...              ...              ...
45588  Metropolitian          (min) 32
45589  Metropolitian          (min) 36
45590  Metropolitian          (min) 16
45591  Metropolitian          (min) 26
45592  Metropolitian          (min) 36

[45593 rows x 20 columns]
```

Now we will perform some feature engineering by converting the latitude and longitude values of the restaurant and delivery location into a distance metric using the geodesic distance. We will also need to format our data accordingly such that the time taken is a float and strip the prefix for the weather conditions column.

```
df["Straight_Line_KM"] = df.apply(lambda x:␣
↪geodesic(np.array([x.Restaurant_latitude, x.
↪Restaurant_longitude]), np.array([x.Delivery_
↪location_latitude, x.Delivery_location_longitude])).
↪km, axis=1)
df["Time_taken(min)"] = df["Time_taken(min)"].
↪apply(lambda x: float(x.split(" ")[-1]))
df["Weatherconditions"] = df["Weatherconditions"].
↪apply(lambda x: x.split(" ")[-1])
```

Next, we will need to drop columns that would not be helpful in predicting the time taken for the delivery to happen.

```
df = df.drop(["Time_Orderd", "Time_Order_picked",
 ↪"Delivery_person_ID", "ID", "Order_Date",
              "Restaurant_latitude", "Restaurant_
 ↪longitude", "Delivery_location_latitude",
              "Delivery_location_longitude"], axis=1)
df = df.drop(df[(df == "NaN ").any(axis=1)].index)
df = df.drop(df[df.Straight_Line_KM > 100].index)
```

Here we will format our data into the appropriate types.

```
df["Vehicle_condition"] = df["Vehicle_condition"].
 ↪astype(str)
df["Delivery_person_Age"] = df["Delivery_person_Age"].
 ↪astype(np.int_)
df["Delivery_person_Ratings"] = df["Delivery_person_
 ↪Ratings"].astype(np.float16)
```

Non-numerical variables will need to be converted into dummy variables to indicate the presence or the absence of a particular condition. For example, in sunny weather, we will have a True value in the Sunny column and False in the Windy, Stormy, Cloudy, Fog, and Sandstorms columns.

```
df = pd.get_dummies(df)
```

Now we will need to format our dataset into features and labels, as well as split it into training and testing.

```
X = df.drop(["Time_taken(min)"], axis=1)
y = df["Time_taken(min)"]
x_train, x_test, y_train, y_test = train_test_split(X,
 ↪ y, test_size=0.33, random_state=42)
```

Let us train our model and evaluate its performance.

```
regressor=RandomForestRegressor(n_estimators= 20,
                                min_samples_split= 4,
                                min_samples_leaf= 1)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

```
r2_score(y_test, y_pred)
```

```
0.8251718756925756
```

```
mean_absolute_error(y_test, y_pred)
```

```
3.105488065872512
```

Great, we were able to get a high regression score of 0.825 and our model has an average error of 3.1 minutes in prediction.

### *12.3.4 Delivery Optimization*

Delivery optimization helps identify the most time and cost-effective way to get from point A to point B by choosing the most effective delivery routes.

More complex optimization algorithms help businesses fill orders while taking into account things like driver schedules, available hours, total stops, fulfillment estimates, and legal requirements. The fastest or cheapest route is not the focus of the optimization. Finding a path that effectively and adequately takes into account all of the relevant factors is the goal.

In this example, we will demonstrate how we can optimize supply chains by determining how many croissants should be delivered to each café from each bakery in Manhattan [6]. We will achieve this by minimizing the Wasserstein loss, otherwise known as the Earth mover's distance.

We will be using the Python Optimal Transport package for this task [6].

```
try:
    import ot
except:
    !pip install POT
```

```
import numpy as np
import matplotlib.pyplot as plt
import time
import ot
```

The data that we are using was extracted through a Google maps query for bakeries and cafés in Manhattan. Fictional production and sales values were synthesized that both add up to the same number of croissants. The file can be downloaded from [6]:

https://github.com/PythonOT/POT/blob/master/data/manhattan.npz?raw=true

In this case the solution will be generated by finding the Earth mover's distance. The Earth mover's distance is the minimum cost incurred to convert one distribution (production of croissants) into another (consumption of croissants) when provided a cost matrix.

```
data = np.load('Supply_Chain/manhattan.npz')

bakery_pos = data['bakery_pos']
bakery_prod = data['bakery_prod']
cafe_pos = data['cafe_pos']
cafe_prod = data['cafe_prod']
Imap = data['Imap']

print('Bakery production: {}'.format(bakery_prod))
print('Cafe sale: {}'.format(cafe_prod))
print('Total croissants : {}'.format(cafe_prod.sum()))
```

```
Bakery production: [31. 48. 82. 30. 40. 48. 89. 73.]
Cafe sale: [82. 88. 92. 88. 91.]
Total croissants : 441.0
```

Let us plot the location and the quantities supplied/demanded on the map. Larger circles will represent larger quantities demanded/supplied.

```
plt.figure(1, (14, 12))
plt.clf()
plt.imshow(Imap, interpolation='bilinear')  # plot␣
↪the map
plt.scatter(bakery_pos[:, 0], bakery_pos[:, 1],␣
↪s=bakery_prod, c='r', ec='k', label='Bakeries')
plt.scatter(cafe_pos[:, 0], cafe_pos[:, 1], s=cafe_
↪prod, c='b', ec='k', label='Cafés')
plt.legend()
plt.title('Manhattan Bakeries and Cafés')
```

```
Text(0.5, 1.0, 'Manhattan Bakeries and Cafés')
```

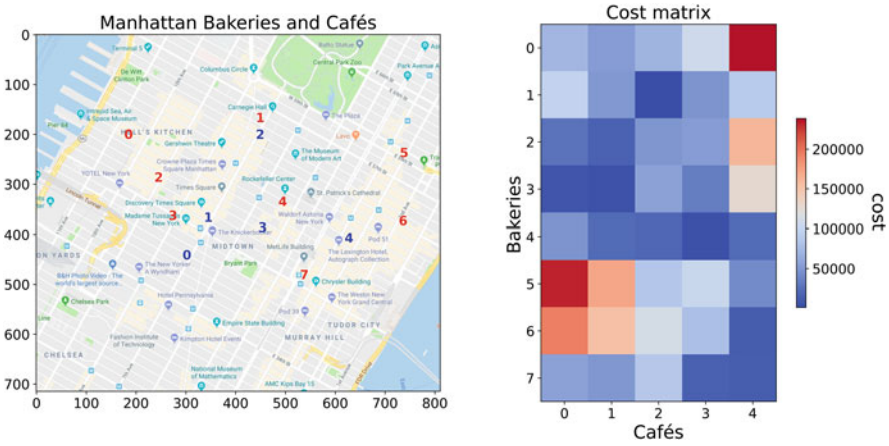

We will now compute the cost matrix between the bakeries and the cafés, which will be the transport cost matrix. The ot.dist will return the squared Euclidean distance between the locations. It can also return other distance metrics such as the Manhattan distance. This cost matrix can also be populated using other methods such as the estimated delivery timings from the earlier example or even a fusion of metrics.

```
C = ot.dist(bakery_pos, cafe_pos)

labels = [str(i) for i in range(len(bakery_prod))]
f = plt.figure(2, (14, 7))
plt.clf()
plt.subplot(121)
plt.imshow(Imap, interpolation='bilinear')  # plot␣
↪the map
for i in range(len(cafe_pos)):
    plt.text(cafe_pos[i, 0], cafe_pos[i, 1],␣
↪labels[i], color='b',
```

(continues on next page)

(continued from previous page)

```
            fontsize=14, fontweight='bold', ha='center
↪', va='center')
for i in range(len(bakery_pos)):
    plt.text(bakery_pos[i, 0], bakery_pos[i, 1],␣
↪labels[i], color='r',
            fontsize=14, fontweight='bold', ha='center
↪', va='center')
plt.title('Manhattan Bakeries and Cafés')

ax = plt.subplot(122)
im = plt.imshow(C, cmap="coolwarm")
plt.title('Cost matrix')
cbar = plt.colorbar(im, ax=ax, shrink=0.5, use_
↪gridspec=True)
cbar.ax.set_ylabel("cost", rotation=-90, va="bottom")

plt.xlabel('Cafés')
plt.ylabel('Bakeries')
plt.tight_layout()
```




The red cells in the matrix image show the bakeries and cafés that are further away and thus more costly to transport from one to the other, whereas the blue ones show those that are closer to each other, with respect to the squared Euclidean distance.

Now let us find the solution by using ot.emd.

```
start = time.time()
ot_emd = ot.emd(bakery_prod, cafe_prod, C)
time_emd = time.time() - start
print(time_emd, "seconds")
```

```
0.0009844303131103516 seconds
```

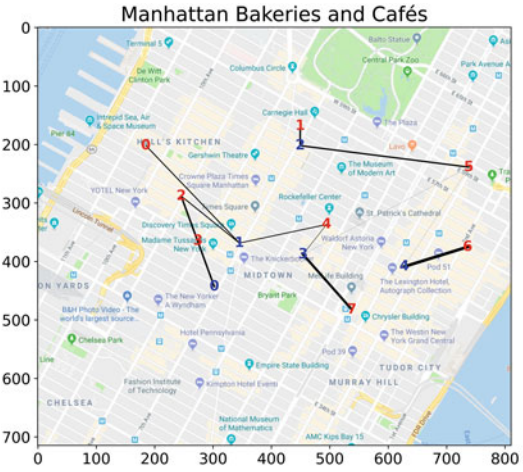
Now let us visualize the solution. The thick lines should show the larger amounts of croissants that need to be transported from one bakery to another.

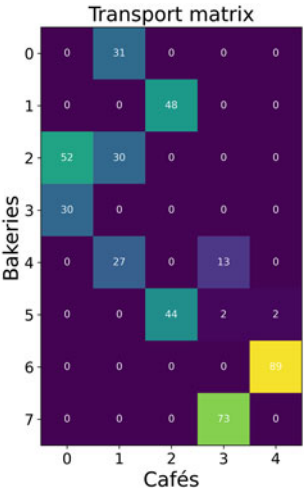
```
# Plot the matrix and the map
f = plt.figure(3, (14, 7))
plt.clf()
plt.subplot(121)
plt.imshow(Imap, interpolation='bilinear')  # plot␣
↪the map
for i in range(len(bakery_pos)):
    for j in range(len(cafe_pos)):
        plt.plot([bakery_pos[i, 0], cafe_pos[j, 0]],␣
↪[bakery_pos[i, 1], cafe_pos[j, 1]],
                 '-k', lw=3. * ot_emd[i, j] / ot_emd.
↪max())
for i in range(len(cafe_pos)):
    plt.text(cafe_pos[i, 0], cafe_pos[i, 1],␣
↪labels[i], color='b', fontsize=14,
             fontweight='bold', ha='center', va='center
↪')
for i in range(len(bakery_pos)):
    plt.text(bakery_pos[i, 0], bakery_pos[i, 1],␣
↪labels[i], color='r', fontsize=14,
             fontweight='bold', ha='center', va='center
↪')
plt.title('Manhattan Bakeries and Cafés')

ax = plt.subplot(122)
im = plt.imshow(ot_emd)
for i in range(len(bakery_prod)):
    for j in range(len(cafe_prod)):
        text = ax.text(j, i, '{0:g}'.format(ot_emd[i,␣
↪j]),
                       ha="center", va="center",␣
↪color="w")
plt.title('Transport matrix')
```

(continues on next page)

(continued from previous page)

```
plt.xlabel('Cafés')
plt.ylabel('Bakeries')
plt.tight_layout()
```




We can calculate the minimized loss value for comparison as well.

```
W = np.sum(ot_emd * C)
print('Wasserstein loss (EMD) = {0:.2f}'.format(W))
```

```
Wasserstein loss (EMD) = 10838179.41
```

## Exercises

1. List three different roles and responsibilities of supply chain departments.
2. For each stage of the supply chain cycle below, explain how making improvements to each stage will impact the supply chain.
   - Planning
   - Sourcing
   - Manufacturing
   - Delivering
   - Returning

3. List three examples of supply chain models and state its benefits.
4. Define the bullwhip effect.
5. List three potential reasons for variations in quantity ordered.
6. Provide three solutions to reduce the bullwhip effect.
7. Identify three different challenges in supply chain management.
8. For each of the challenges listed in 7, identify one way that artificial intelligence could potentially be used to alleviate the problem.
9. Perform demand forecasting using google trends data for a few products:

   - Beanies
   - Tote bags
   - Bicycles

   Observe the differences between the trends of the products and determine what kind of trend and seasonality suits them better.

## References

1. Ashley M (2022) Understanding the Supply Chain Bullwhip Effect. https://www.truecommerce.com/blog/bullwhip-effect-supply-chain. Accessed 27 Apr 2023
2. Brutlag J (2000) Aberrant behavior detection in time series for network monitoring. In: Proceedings of the 14th USENIX conference on System administration, pp 139–146
3. Gaurav M (2022) Food delivery dataset. In: Kaggle. https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset. Accessed 27 Apr 2023
4. Jason F (2022) Supply Chain Management (SCM): How It Works and Why It Is Important. https://www.investopedia.com/terms/s/scm.asp. Accessed 27 Apr 2023
5. ravirajsinh45 (2020) casting product image data for quality inspection. https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting-product. Accessed 27 Apr 2023
6. Rémi F, Nicolas C (2016) Introduction to Optimal Transport with Python & —POT Python Optimal Transport 0.8.2 documentation. https://pythonot.github.io/auto_examples/plot_Intro_OT.html#sphx-glr-auto-examples-plot-intro-ot-py. Accessed 27 Apr 2023
7. Shehmir J (2023) Top 12 AI Use Cases for Supply Chain Optimization in 2023. https://research.aimultiple.com/supply-chain-ai/. Accessed 27 Apr 2023
8. Stitchdiary (2017) The Bullwhip Effect. https://medium.com/@stitchdiary/the-bullwhip-effect-c40751d768ba. Accessed 27 Apr 2023

# Chapter 13
# AI in Operations Management


**Learning Outcomes**

- Understand the roles and responsibilities of operations departments.
- Explain the steps performed in business process management.
- Explain the steps performed in Six Sigma.
- Identify ways that artificial intelligence can be applied to operations management.
- Develop artificial intelligence solutions for predicting root causes, predicting machine failures, and performing process automation.

## 13.1 Introduction to Operations Management

Operations management (OM) is the management of business execution in order to achieve the highest level of planning within an organization. It is concerned with converting materials and labor into goods and services as efficiently as feasible in order to increase an organization's profit. The operations management team strives to achieve the maximum net operational profit feasible by balancing costs and income.

Operations management is concerned with the use of assets such as workers, resources, appliances, and technology. Operations managers acquire, expand, and supply commodities to clients based on client wants and the company's capabilities.

Operations management is responsible for a variety of critical organizational decisions. Some examples are determining the scale of manufacturing factories, project administration procedures, and maintaining the information technology infrastructure. Other operational difficulties include inventory level management, which includes business process levels and raw material purchases, quality control, manual handling, and maintenance standards.



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_13

### *13.1.1 Business Process Management*

Business process management is a practice where you evaluate, optimize, and automate your business processes [2]. Business process management must be done regularly as processes in order to continuously improve processes to increase the operational efficiency.

The lifecycle of business process management includes five steps as shown in Fig. 13.1:

1. Design and analyze ideas for how to improve and optimize the process.
2. Model how the changes would impact the process and the entire organization.
3. Execute your new processes using software that allows you to easily create and edit them.
4. Monitor and control your processes to ensure that they are working properly.
5. Optimize and refine your implementation continuously.

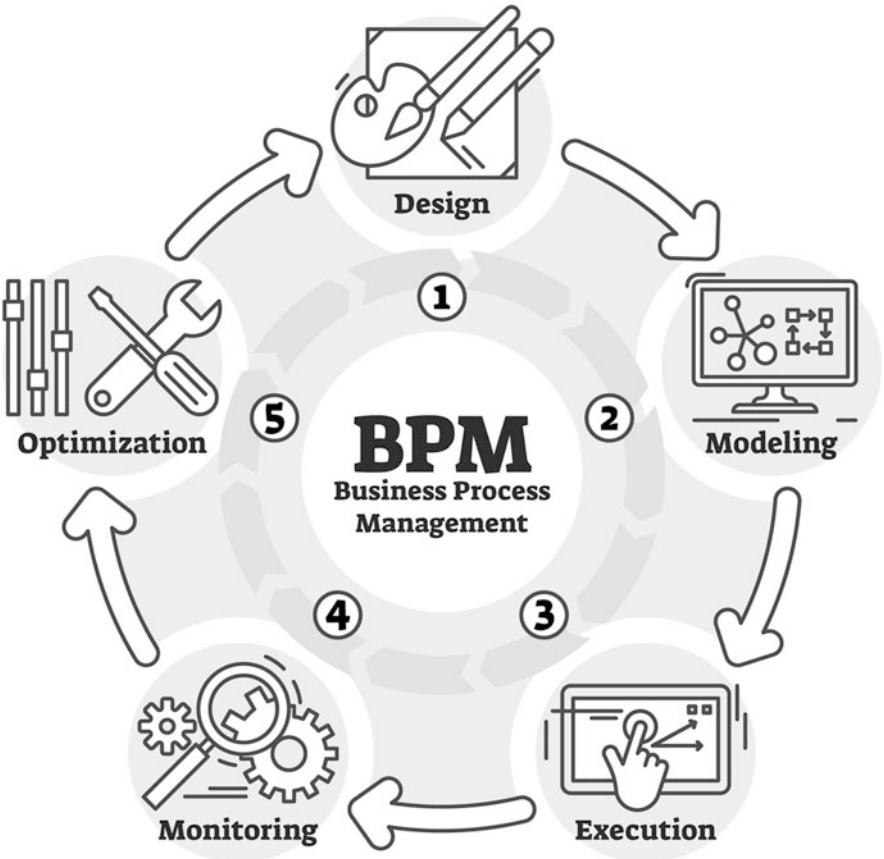



**Fig. 13.1** Business process management [5]

In certain instances, it is preferable to re-engineer your business processes rather than making incremental improvements. This implies that you are beginning over by recreating your business procedures from scratch. Sometimes, new technologies can help to simplify this task and make it easier than current operational processes.

### *13.1.2 Six Sigma*

Six Sigma is a management method to help improve the quality of goods and services by identifying issues with the current processes. Six Sigma aims to achieve less than 3.4 inadequacies or inefficiencies across one million opportunities.

DMAIC is the most widely used tool for implementing the Six Sigma methodology. It works to improve manufacturing processes in 5 easy steps as outlined in Fig. 13.2:

Define – Measure – Analysis – Improvement – Control

- **Define**: Determine the issue with a certain manufacturing process and how it might be improved. Identify the appropriate tools or resources to employ.
- **Measure**: Quantify the performance of the process. This could help to figure out the best way forward to improve performance.
- **Analysis**: Analyze the process to find out what was done wrong or could be improved.
- **Improvement**: Identify solutions and determine the best option forward.
- **Control**: Make small changes across each new process so that results can be easily comparable against previous results to determine if things are getting better or not.

### *13.1.3 Supply Chain Management (SCM) vs. Operations Management (OM)*

Supply chain management and operations management are two areas that are closely related, and some executives in both fields are well-versed in both [4]. There are discrepancies in what each type of executive handles, including the fact that they supervise different parts of the organization. If you are strong at organizing, transmitting, and managing information, either field could be a good fit for you.

The following are some of the differences between supply chain management and operations management [4]:

- **Management of Force vs. Management of Functions**

  Supply chain management is mostly about managing the flow of goods and services. Operations management, on the other hand, is mostly about managing operations or functions.


**Fig. 13.2** Six sigma [6]

- **Concerns Within Company vs. Outside Company
  Supply chain management is primarily concerned with events that occur outside of the company or trade, such as the delivery of products to appropriate areas, the acquisition of materials, and so on, whereas operations management is primarily concerned with what occurs within a business or trade.
- **Supply vs. Process**
  Supply chain management manages the supply or movement of goods and products, whereas operations management manages the process of goods and products.
- **Evaluation vs. Planning**
  The majority of supply chain management time is spent on appraising, which is analyzing and approving distributors based on qualitative and quantitative criteria, and on negotiable contracts, which is creating contracts between parties.

In contrast, the majority of time in operations management is spent planning, directing, arranging, and executing.

- **Supply Chain Activities vs. Development Activities**
  The supply chain management process involves the design, planning, execution, control, and monitoring of all supply chain activities. The operation management process includes planning, coordinating, and directing work, which leads to increased development.
- **Reduction in Operating Costs vs. Increase in Revenue**
  Supply chain management benefits include lower operational expenses, increased cash flow, reduced risk, improved quality control, and so on. Operations management benefits include increased revenue, engaged employees, improved customer happiness, improved product quality, and so on.
- **Difference in Objectives**
  The primary goals of supply chain management are to increase the total value generated, improve overall organizational performance, effectively regulate inventories, and so on, whereas the primary goals of operations management are leveraging organizational resources to develop products, services, or goods, boosting customer satisfaction, and providing things of desired quality on schedule.

## 13.2 Artificial Intelligence in Operations Management

Artificial intelligence can be used to increase the efficiency of business operations in many ways through efficiently utilizing data for predicting events. This can help operation managers reduce the amount of repetitive work done by staff or deliver insights that can help reduce costs in the organization. Some examples of how artificial intelligence can be applied to business operations are as follows:

1. **Predicting operational failures**: This can help flag out aberrations and allow the organization to avoid critical catastrophes. Such AI-enabled systems should be able to monitor the materials, machines, and equipments in order to predict potential failures. Furthermore, they can also be extended to systems that interact with customers such as orders and supply–procurement.
2. **Automating repetitive processes**: As a result of outsourcing the mundane to "smart" technologies, the workers can focus on more value-adding jobs and the unit can run more efficiently with fewer personnel.
3. **Predict causes of failures**: Common issues that often occur can be fed into these intelligent algorithms that can help recommend the best way to tackle incidences that occur during operations.
4. **Improving product quality checks**: Product defects in the manufacturing process can be reduced through the use of AI to flag out anomalies in the quality checking process by utilizing past data to train models to detect defects, thus improving the quality of the manufacturing process.

## 13.3 Applications of AI in Operations

### *13.3.1 Root Cause Analysis for IT Operations*

Root cause analysis is an iterative, questioning strategy used to investigate the cause-and-effect links of a fundamental problem.

When a problem occurs, we are only able to observe the problem's symptoms. The symptoms must be mapped to a root cause by asking a series of exploratory questions and executing a battery of diagnostic tests. This is comparable to how a physician links a symptom, such as a fever, to its underlying cause, such as a viral infection.

IT operations receive numerous service incidents from users every day. These incidences usually describe only the symptoms seen by the user. IT operation engineers will need to investigate the issue further and uncover more symptoms. This process requires analyzing the symptoms in order to identifying the underlying cause as specifically as possible.

For example, symptoms might be a user observing a message stating, "We are unable to save your changes. Please contact the administrator." This might be due to having no disk space in the database. Time is required for the IT operations staff to find out the root cause, and this process of investigation can impact resolution times significantly. This may require the help of other experts, which could be the developers or the product vendor which could lengthen resolution times. AI can help here by observing the symptoms and helping to make a data-driven guess on what the root cause might be, thus helping staff to fix the root causes faster. In this example, we will build an AI model that can help predict the root cause of the problems faced using features extracted from logs.

The data we will be using is called root_cause_analysis.csv. It can be obtained from [1]:
https://www.kaggle.com/datasets/aryashah2k/datasets-in-it-ops-applied-ai.

This data describes the symptoms observed for various problems reported by users and are gathered by IT operation engineers. Each symptom is a separate attribute with values containing either one if the symptom is seen or zero when it is not. The symptoms are CPU load, memory load, delays, and whether some of these specific error codes were seen in the log files. The root cause attribute is the target variable that contains classes of root causes.

This example uses a simple set of symptoms. However, in real-life scenarios, there can be numerous symptoms. Symptoms can be further classified based on where they are observed. For example, was the CPU load at the load balancer or a single micro-service? Text mining can also be used to extract symptoms from broad text.

We will begin by loading up the dataset and preprocess it to make it ready for machine learning consumption.

First we load the CSV into a Pandas dataframe using the read_csv method. We then print the data types and contents of the dataframe to check if the loading happened successfully.

Next, we need to convert data to formats that can be consumed. In this case, the root cause is in text which needs to be converted into an integer. We will use the label encoder from Scikit-learn to transform the root cause into an integer. We then separate the features into X and labels into Y.

```
# Importing necessary libraries
import pandas as pd
import os
import tensorflow as tf

#Load the data file into a Pandas Dataframe
symptom_data = pd.read_csv("Operations/root_cause_
↪analysis.csv")

#Explore the data loaded
print(symptom_data.dtypes)
symptom_data.head()
```

```
ID              int64
CPU_LOAD        int64
MEMORY_LOAD     int64
DELAY           int64
ERROR_1000      int64
ERROR_1001      int64
ERROR_1002      int64
ERROR_1003      int64
ROOT_CAUSE     object
dtype: object
```

```
   ID  CPU_LOAD  MEMORY_LOAD  DELAY  ERROR_1000  ERROR_1001  ERROR_1002  \
0   1         0            0      0           0           1           0
1   2         0            0      0           0           0           0
2   3         0            1      1           0           0           1
3   4         0            1      0           1           1           0
4   5         1            1      0           1           0           1

   ERROR_1003     ROOT_CAUSE
0           1         MEMORY
1           1         MEMORY
2           1         MEMORY
3           1         MEMORY
4           0  NETWORK_DELAY
```

```
# Convert input data to formats that can be used by␣
↪ML Algorithms
from sklearn import preprocessing
```

(continues on next page)

(continued from previous page)

```
label_encoder = preprocessing.LabelEncoder()
symptom_data['ROOT_CAUSE'] = label_encoder.fit_
↪transform(
                                    symptom_data['ROOT_
↪CAUSE'])

#Convert Pandas DataFrame to a numpy vector
np_symptom = symptom_data.to_numpy()

#Extract the feature variables (X)
X = np_symptom[:,1:8]

#Extract the target variable (Y)
Y = np_symptom[:,8]

print("Shape of feature variables :", X.shape)
print("Shape of target variable :",Y.shape)
```

```
Shape of feature variables : (1000, 7)
Shape of target variable : (1000,)
```

Let us create a decision tree model to predict the root cause based on the symptoms. Decision tree models allow us to trace the various conditions that may lead to the error observed. To illustrate this with a simpler example, we will set the max_depth to 3 and visualize the trained model.

For testing, we will set aside 20% of the records to be used to evaluate our models.

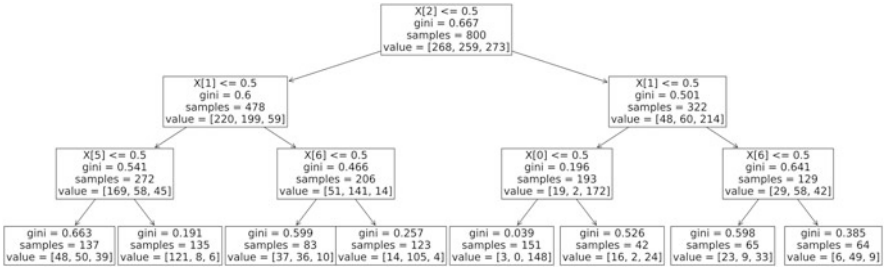
```
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,␣
↪cross_val_score
from sklearn.tree import DecisionTreeClassifier

X_train, X_test, y_train, y_test = train_test_split(X,
↪ Y, test_size=0.20, random_state=42)
clf = DecisionTreeClassifier(random_state=0, max_
↪depth = 3)
clf.fit(X_train, y_train)
print("Accuracy score: {:.2f}%".format(clf.score(X_
↪test, y_test) * 100))
```

(continues on next page)

(continued from previous page)

```
plt.figure(0, figsize=(30, 10))
tree.plot_tree(clf, fontsize=20)
plt.show()
```

```
Accuracy score: 73.50%
```




With a depth of 3, the tree is able to isolate certain cases of NETWORK DELAYS very well. In the leaf node where the gini coefficient is 0.039 and the training samples containing classes of [3, 0, 148], these cases are defined when DELAY is True, but both CPU_LOAD and MEMORY_LOAD are False. This makes sense as the error is not caused by the computer, but rather a network issue.

In the example above, we restricted the model to only a depth of 3 for simple visualization. But because of that, it can only look at 3 binary features. We will now remove the restriction and perform k-fold cross validation to estimate the expected accuracy of the trained model. K-fold cross validation trains 5 models with 5 different splits of the data. Once it is done, we can calculate the average accuracy over the 5 models for a reliable estimate of the model's performance when deployed.

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=0)
print("Accuracy score: {:.2f}%".format(cross_val_
↪score(clf, X, Y, cv=5).mean() * 100))
```

```
Accuracy score: 84.10%
```

We see that the model can achieve an accuracy of 84.10% with the test data. Now that we have the model setup, let us train it with our split.

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=0)
```

(continues on next page)

(continued from previous page)

```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy score: {:.2f}%".format(clf.score(X_
↪test, y_test) * 100))
```

```
Accuracy score: 86.00%
```

Our current model predicts 86% of the test set correctly. Now we will demonstrate how this model can be used to predict root causes.

Now that we have developed a Root Cause Analysis model, we can use it to predict the cause for a new incident. When a new incident happens, we can identify the symptoms of the incident and populate the related feature variables here, like CPU load, memory load, delays, and error codes.

These features are then passed to the model's prediction function which will return a class prediction for the root cause. The class will then be translated into the underlying root cause, wherein this case would be either DATABASE_ISSUE, MEMORY, or NETWORK_DELAY.

In the below example, we demonstrate that when CPU_LOAD is high and when ERROR_1003 appears, it is likely due to a MEMORY issue. This could help IT operations staff quickly try to isolate the problem by looking into the server's memory.

```
CPU_LOAD=1
MEMORY_LOAD=0
DELAY=0
ERROR_1000=0
ERROR_1001=0
ERROR_1002=0
ERROR_1003=1

prediction = clf.predict([[CPU_LOAD, MEMORY_LOAD,␣
↪DELAY, ERROR_1000, ERROR_1001, ERROR_1002, ERROR_
↪1003]])
print(label_encoder.inverse_transform(prediction))
```

```
['MEMORY']
```

Through this example, we have demonstrated how AI can be used to help identify the root causes of issues in IT operations. This can be further improved by monitoring more features of the system and feeding in more accurate features that can better describe the issues faced by the system.

Improved root cause analysis could potentially help speed up the time taken to resolve issues in IT operations as staff are able to identify and fix the problem quicker.

### *13.3.2 Predictive Maintenance*

Predictive maintenance software employs data science and predictive analytics to forecast when a piece of equipment will break, allowing remedial maintenance to be arranged before the problem occurs. The goal is to plan maintenance at the most convenient and cost-effective time, allowing the equipment's lifespan to be maximized to the greatest extent possible, but before the equipment is impaired.

Predictive maintenance solutions often have an underlying architecture that includes data acquisition and storage, data transformation, condition monitoring, asset health evaluation, prognostics, a decision support system, and a human interface layer.

Nondestructive testing methods such as acoustic, corona detection, infrared, oil analysis, sound level measurements, vibration analysis, and thermal imaging predictive maintenance, which measure and gather operations and equipment real-time data via wireless sensor networks, are examples of predictive maintenance technologies. These measures and predictive maintenance machine learning techniques, such as the classification approach or the regression approach, are used by predictive maintenance service providers to discover equipment weaknesses.

In this example, we will develop a predictive maintenance model for the Air Pressure system (APS) in Scania trucks. The dataset can be obtained here [7]: https://www.kaggle.com/datasets/uciml/aps-failure-at-scania-trucks-data-set.

This collection is made up of data obtained from large Scania vehicles in normal use. The system under consideration is the Air Pressure System (APS), which produces pressured air for use in various truck tasks such as braking and gear changes. The datasets' positive class consists of component failures for a given APS system component. Trucks in the negative class have failed for reasons unrelated to the APS. The data is a subset of all accessible data chosen by specialists.

For proprietary reasons, the data attribute names have been anonymized. It is made up of both single-number counters and histograms made up of bins with varying conditions. Histograms typically feature open-ended conditions at each end. For example, if we measure the ambient temperature "T," the histogram might be characterized as having four bins.

The properties are as follows: class, anonymized operational data, and so on. The operational data has an identifier and a bin id, such as "Identifier Bin." There are 171 attributes in all, 7 of which are histogram variables. "na" represents missing values.

The steps already applied for preprocessing the data in aps_failure_training_set_processed_8bit.csv and aps_failure_test_set_processed_8bit.csv are as follows:

1. Replace missing fields such as "na" with NaN.
2. Map binary-valued column "class" to $\{-1,1\}$ instead of 1 and 0 for neural network models that use the hyperbolic tangent function, tanh(), as the activation function.
3. Fill missing fields using a K-nearest neighbors estimate of the fancyimpute Python package for training and test data separately.

4. Normalize feature columns to zero mean, unit variance in training data. Apply normalization coefficients calculated for training data to test data.
5. Clip feature columns to [−1, 1] to dampen the effect of outliers.
6. Lower the precision of the data so that each value can fit in a single byte. Before this step, all fields were in the [−1, 1] range and were of np.float32 dtype. After this step, they are in the [−0.9921875, 0.9921875] range.

Let us begin by importing the data.

```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix,␣
↪balanced_accuracy_score
from sklearn.linear_model import LogisticRegression

X_train = pd.read_csv('Operations/aps_failure_
↪training_set_processed_8bit.csv').drop('class',␣
↪axis = 1)
y_train = pd.read_csv('Operations/aps_failure_
↪training_set.csv')['class']
y_train = (y_train == 'pos').astype(np.int)

X_test = pd.read_csv('Operations/aps_failure_test_set_
↪processed_8bit.csv').drop('class', axis = 1)
y_test = pd.read_csv('Operations/aps_failure_test_set.
↪csv')['class']
y_test = (y_test == 'pos').astype(np.int)
```

The company has determined that the cost incurred for calling in an unnecessary maintenance check is $10 and the cost incurred for a truck breakdown is $500. Let us calculate the current cost incurred by the company without implementing a maintenance prediction system.

```
cost_1 = 10
cost_2 = 500

predictions = [0 for i in range(len(y_test))]
tn, fp, fn, tp = confusion_matrix(y_test.tolist(),␣
↪predictions).ravel()
check_all_trucks_cost = fp * cost_1 + fn * cost_2
print("cost of checking all trucks everyday:", check_
↪all_trucks_cost, "dollars")

predictions = [1 for i in range(len(y_test))]
```

(continues on next page)

(continued from previous page)

```
tn, fp, fn, tp = confusion_matrix(y_test.tolist(),␣
↪predictions).ravel()
ignore_breakdowns_cost = fp * cost_1 + fn * cost_2
print("cost of not checking trucks:", ignore_
↪breakdowns_cost, "dollars")

print("minimum cost is", min(check_all_trucks_cost,␣
↪ignore_breakdowns_cost), "dollars per", len(y_test),
↪ "truck-days")
```

```
cost of checking all trucks everyday: 187500 dollars
cost of not checking trucks: 156250 dollars
minimum cost is 156250 dollars per 16000 truck-days
```

According to what we have calculated, the cost incurred by the company would be 156250 dollars for not checking any trucks, fixing them only after a breakdown. Using AI to predict the possibility of a breakdown occurring, we can try to reduce the cost incurred to maintain the truck fleet.

```
model = LogisticRegression(max_iter=100000)
model.fit(X_train, y_train)

print("Test Accuracy: {:.4f}%".format(model.score(X_
↪test, y_test) * 100))
```

```
Test Accuracy: 98.9500%
```

In this case we have developed a model that can predict with 98.95% accuracy. This looks great, but in actual fact, regular accuracy may not represent the data well as there are more cases of no breakdowns than breakdowns. Instead, we should use the balanced accuracy score metric which calculates the accuracy score for each class and averages them so that all classes get an equal weightage.

```
print("Number of operational truck-days:", len([i for␣
↪i in y_test if i == 0]))
print("Number of breakdowns:", len([i for i in y_test␣
↪if i == 1]))
print()
print("Balanced accuracy score: {:.4f}%".
↪format(balanced_accuracy_score(y_test, model.
↪predict(X_test)) * 100))
```

```
Number of operational truck-days: 15625
Number of breakdowns: 375

Balanced accuracy score: 83.5861%
```

```
predictions = model.predict(X_test)
tn, fp, fn, tp = confusion_matrix(y_test.tolist(),␣
↪predictions).ravel()
cost = fp * cost_1 + fn * cost_2

print("cost is", cost, "dollars per", len(y_test),
↪"truck-days")
```

```
cost is 61460 dollars per 16000 truck-days
```

With the current model, we have been able to decrease the cost of operations from 156250 down to 61460 per 16000 truck-days.

Knowing that the data is imbalanced with 59 times more cases of no breakdown than breakdowns, the model would be biased to predict that there is no breakdown. Thus, we can try to reduce this effect by applying SMOTE to create more samples of the under-represented class by creating more synthetic samples through interpolating between features of breakdowns.

```
from imblearn.over_sampling import SMOTE

oversample = SMOTE(random_state=3)
X_train2, y_train2 = oversample.fit_resample(X_train,␣
↪y_train)
```

```
model.fit(X_train2, y_train2)
print("Balanced accuracy score: {:.4f}%".
↪format(balanced_accuracy_score(y_test, model.
↪predict(X_test)) * 100))

tn, fp, fn, tp = confusion_matrix(y_test.tolist(),␣
↪model.predict(X_test)).ravel()
cost = fp * cost_1 + fn * cost_2
print("cost is", cost, "dollars per", len(y_test),
↪"truck-days")
```

```
Balanced accuracy score: 95.3696%
cost is 16470 dollars per 16000 truck-days
```

```
print("cost savings in percent = {:.4f}%".format((1 -␣
↪cost/min(check_all_trucks_cost, ignore_breakdowns_
↪cost)) * 100))
```

```
cost savings in percent = 89.4592%
```

After accounting for class imbalance, we were able to further reduce the cost down to 16470 dollars per 16000 truck-days. Through this example, we have demonstrated that a model trained for predictive maintenance is able to reduce the operational cost by 89.45%.

### *13.3.3 Process Automation*

Operations can be further streamlined through the use of AI by process automation. Instead of having humans to perform mundane and repetitive tasks, we can automate these tasks through the use of robotic process automation.

Robotic process automation (RPA) is a technology that automates digital operations by using robots that mimic human actions. Bots can mimic and learn rule-based actions, as well as interact with any system or program in the same way that humans do. RPA bots, on the other hand, can work around the clock without tiring and do not exhibit bias in procedures like humans. Artificial intelligence is used in RPA to help deal with semi-structured and unstructured data from various media such as text, images, webpages, PDFs, and videos. For example, Optical Character Recognition is one of the most common forms of AI that is applied in RPA to recognize printed and handwritten characters on scanned documents.

As a result, RPA services have become an essential component of business operations. Many corporate operations duties, including marketing, information technology (IT), scheduling, and time tracking, can be automated. The IT system, which is the backbone of any firm, is the biggest benefit of RPA technology.

In this example, we will go through how RPA can be used to automate a form filling process using UiPath [3].

Step 1: Let us begin by creating a csv file containing fake personal details as shown below in Fig. 13.3. The Google forms will be populated with this data.

Step 2: Open your UiPath studio and click on Process as shown in Fig. 13.4.

Step 3: Select a name for your process and click on the Create button as shown in Fig. 13.5.

Step 4: Click on Open Main Workflow as shown in Fig. 13.6.

Step 5: Click on “Drop Activity Here” and then search for “Sequence” activity from the activity box as shown in Fig. 13.7.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Name | Email | Address | | | | |
| 2 | Susan Aniba | anibal.jakub | 199 Fairmont Avenue Kansas City Missouri(MO) 64106 | | | | |
| 3 | Aaron Parke | parker.bin6 | 3516 Hilltop Street Ashfield Massachusetts(MA) 01330 | | | | |
| 4 | Kent Jaquar | jaquan1985 | 4400 Victoria Street Lake Villa Illinois(IL) 60046 | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |

**Fig. 13.3** Step 1: Example excel

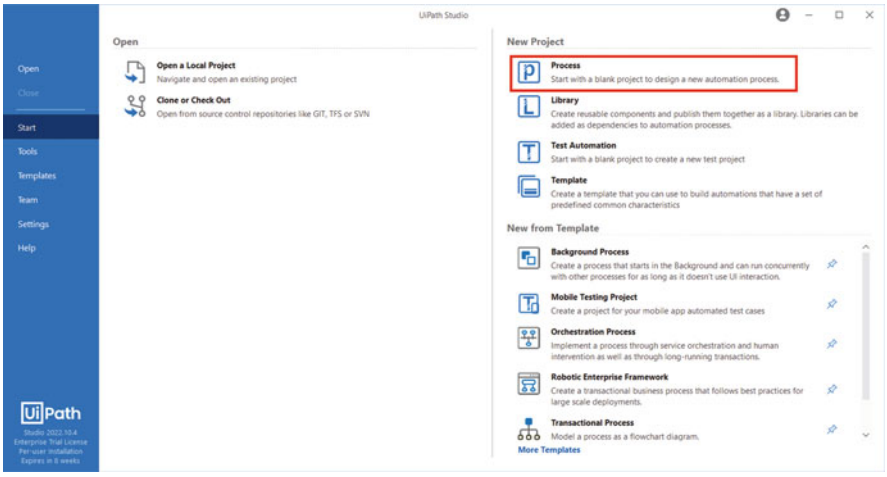



**Fig. 13.4** Step 2: Click on process

Step 6: After selecting a sequence activity, click on "Drop Activity Here," search for Read CSV activity, and add it as shown in Fig. 13.8. The Read CSV activity is used for reading the CSV file.

Step 7: Click on the folder icon to provide the address of your csv file. Then create a new output variable by clicking on the "+" icon followed by clicking on "Create Variable" as shown in Fig. 13.9. Check the csv header file so that we can use the csv headers such as Name, Address, and Contact.

Step 8: Name the variable as data and press Enter as shown in Fig. 13.10.

Step 9: Now add For Each Row activity by clicking on the "+" button and searching for it in the search bar as shown in Fig. 13.11. This will allow us to process every row of the CSV file line by line.

Step 10: Under "ForEach," provide an iteration variable named CurrentRow as shown in Fig. 13.12. Under 'In," click on the "data" variable that we have defined earlier from our read_csv process.

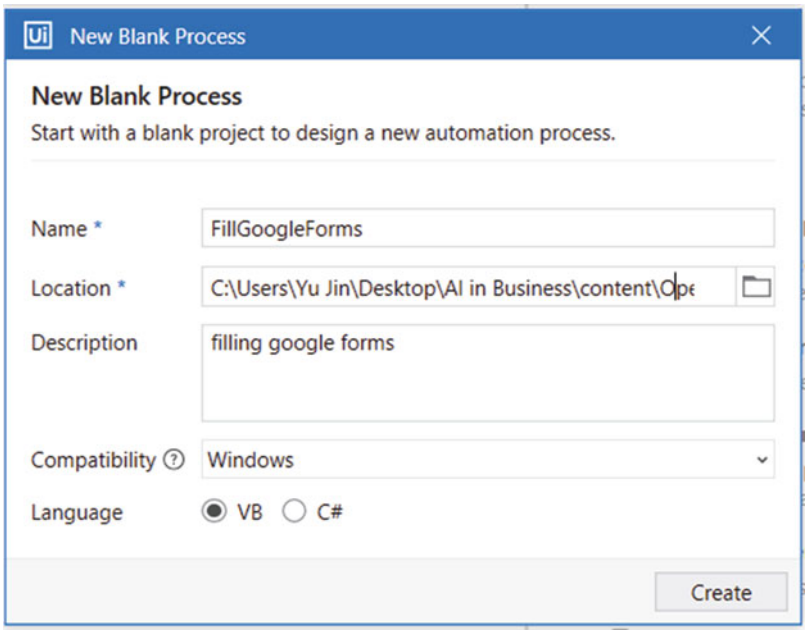



**Fig. 13.5** Step 3: Create process

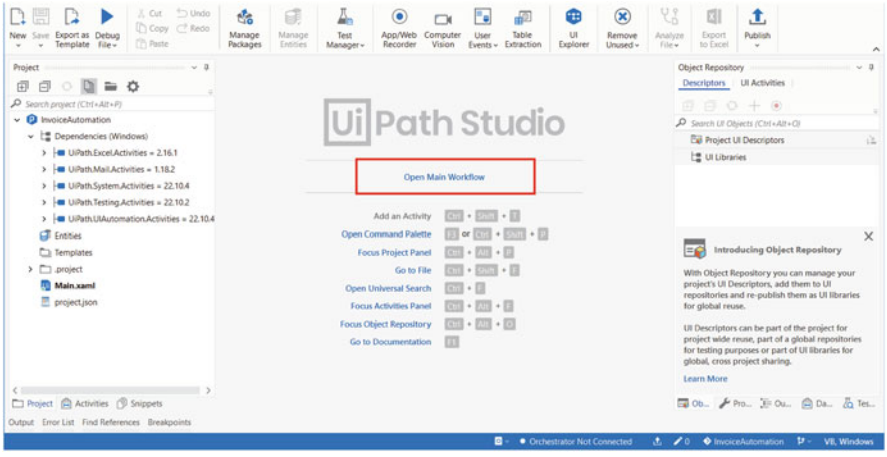



**Fig. 13.6** Step 4: Open workflow

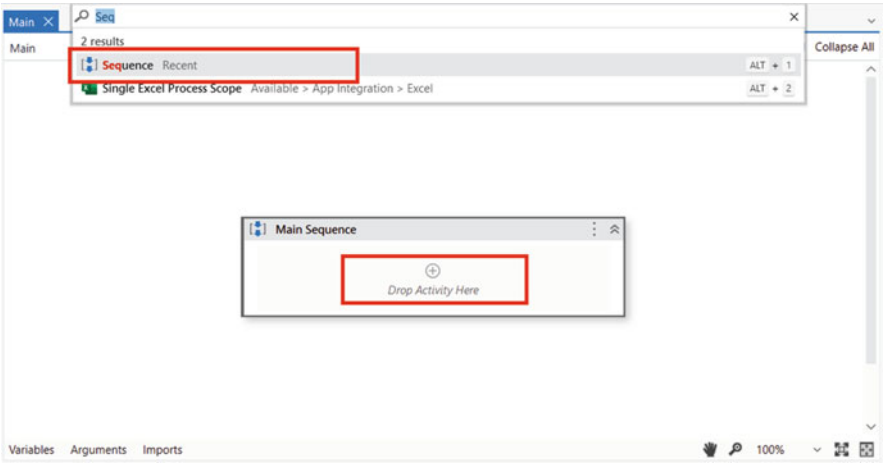



**Fig. 13.7** Step 5: Create sequence activity

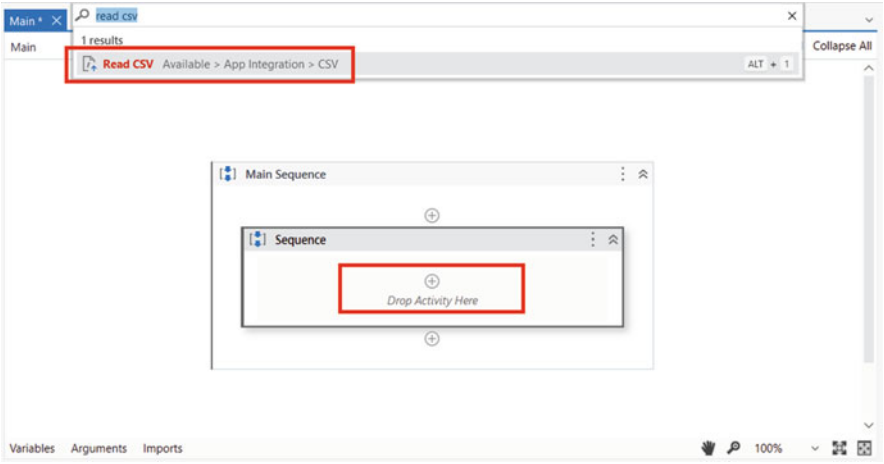



**Fig. 13.8** Add Read CSV operation

Step 11: Now open up the link to the Google form you want to fill in a browser of your choice as shown in Fig. 13.13.

Step 12: Now in the body of each row activity, click on the "+" button and add the Use Application/Browser activity as shown in Fig. 13.14. This will allow us to open our browser.

Step 13: Click on the "Indicate application to automate" button as shown in Fig. 13.15.

Step 14: Provide the Google form link under Browser URL. Click on the "+" sign, search for the "type into" activity, and add it to the sequence as shown in

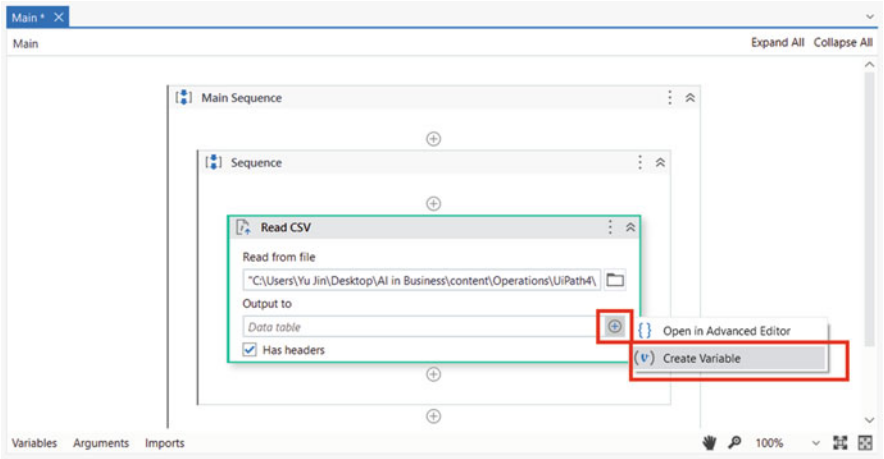



**Fig. 13.9** Step 7: Create variables

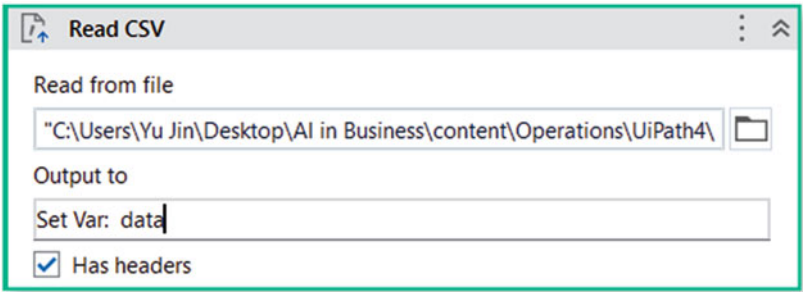



**Fig. 13.10** Step 8: Name variable

Fig. 13.16. In this example we will create 3 sets of type into activities because I have 3 fields in my Google form, but you can add as many according to your requirement.

Step 15: First make sure that your Google form opened in the background of your UiPath and after that click on "Indicate in Edge" (Fig. 13.17).

Step 16: Click on the field on the webpage where you want to type/ add data into and anchor it to something descriptive as shown in Fig. 13.18.

Step 17: Type "CurrentRow("Name").toString" to reference the Name variable in the csv file as shown in Fig. 13.19. Repeat steps 14 to 17 for all the fields required for the "Type Into Activities" actions.

Step 18: Now search for "Click Activity" and add it as shown in Fig. 13.20.

Step 19: Anchor the click to the submit button by clicking once on the button and another time on the "Submit" word as shown in Fig. 13.21.

Step 20: Click on the "+" button and search for "Navigate Browser". Add it to the sequence as shown in Fig. 13.22.

Step 21: Select "Close Tab" to close the form as shown in Fig. 13.23.

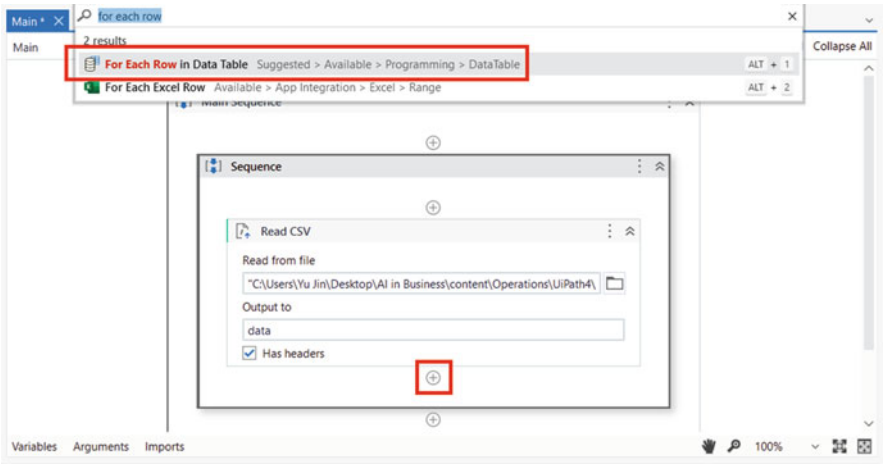



**Fig. 13.11** Step 9: Iterate through rows

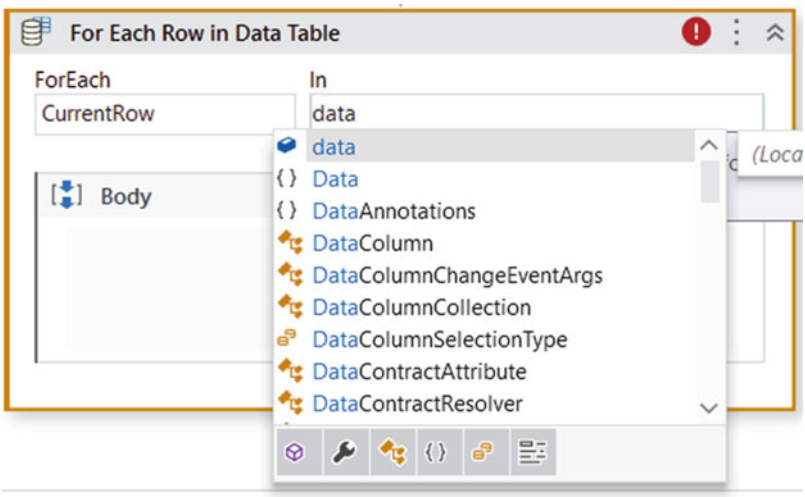



**Fig. 13.12** Step 10: Iterate through rows

Step 22: Now our Process is Ready and you just Run by clicking on the green "Play" button. The RPA script will read all the rows available in our CSV file and fill them all into the Google Form one by one without any human interaction needed.

This example demonstrates how RPA can be used to automate repetitive tasks such as filling details into Google forms. There are many other useful applications of RPA such as invoice automation, employee onboarding, payroll automation, and many others. The automation of various business operations can help reduce the amount of tasks done by humans and therefore increasing operational efficiency across the business.

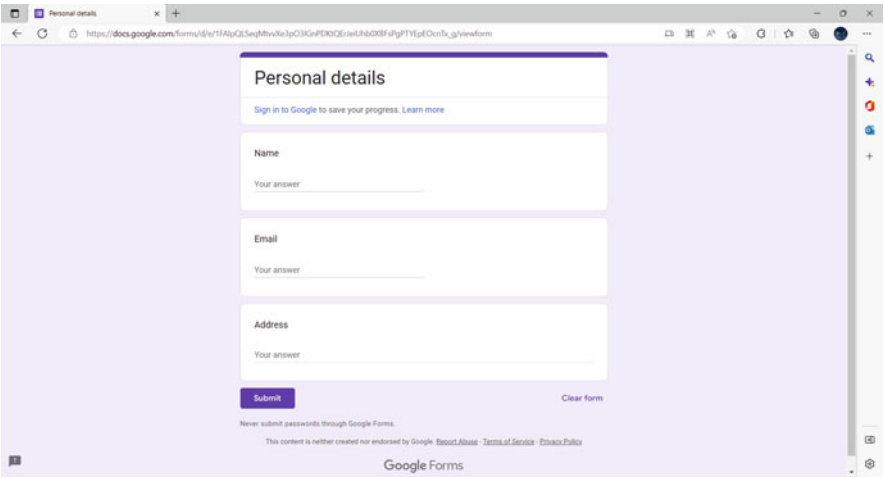



**Fig. 13.13** Step 11: Open webpage

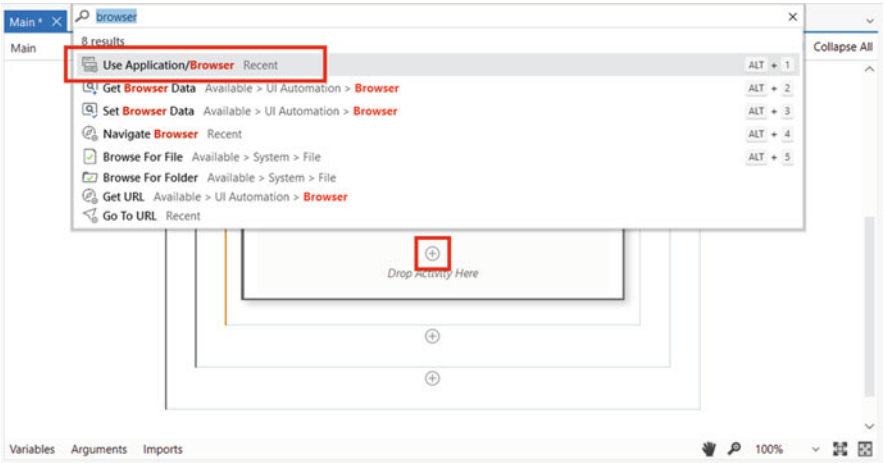



**Fig. 13.14** Step 12: Open webpage

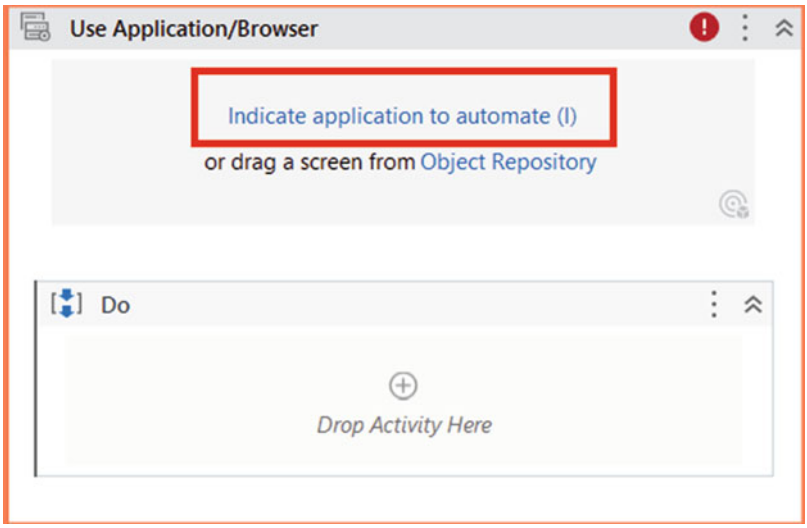



**Fig. 13.15** Step 13: Select application to automate

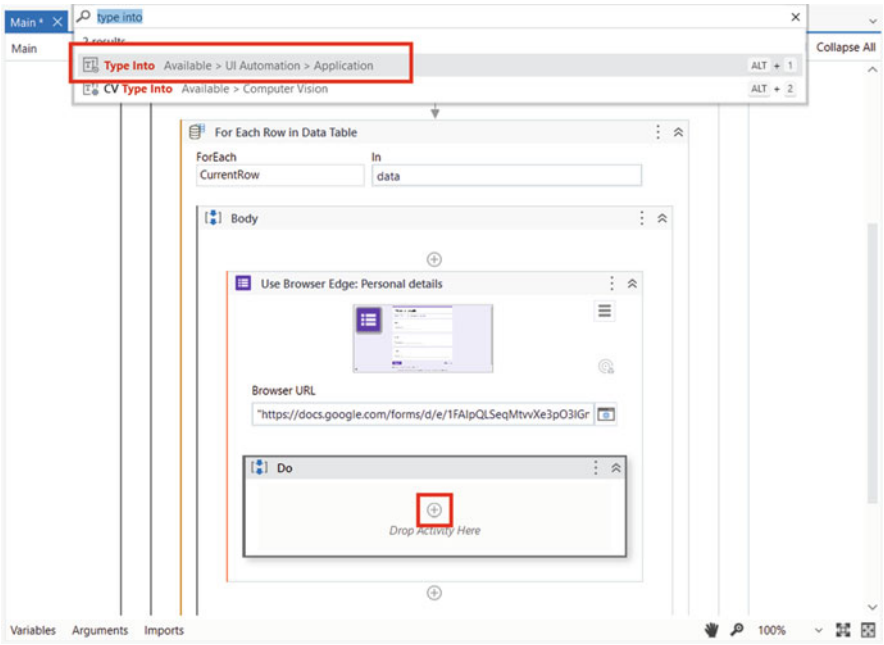



**Fig. 13.16** Step 14: Set up automatic typing

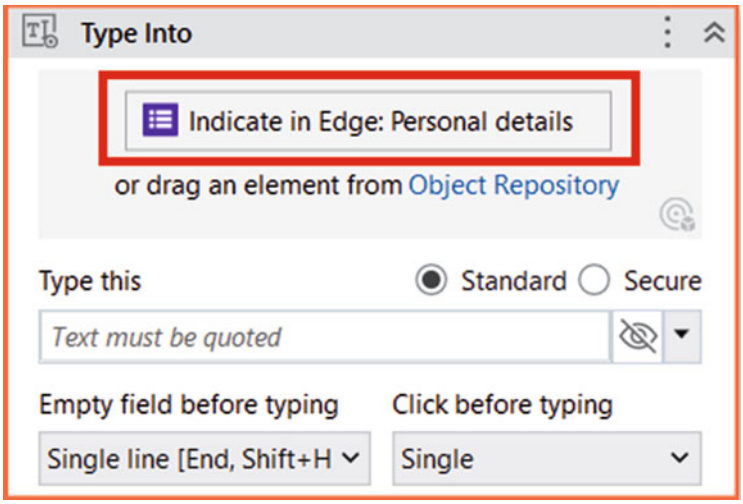



**Fig. 13.17** Step 15: Click on indicate in Edge

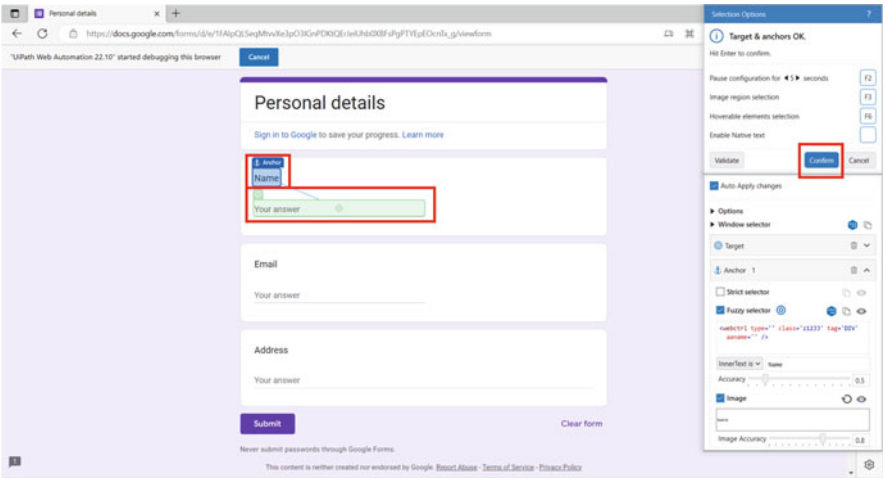



**Fig. 13.18** Step 16: Click on anchors in the webpage

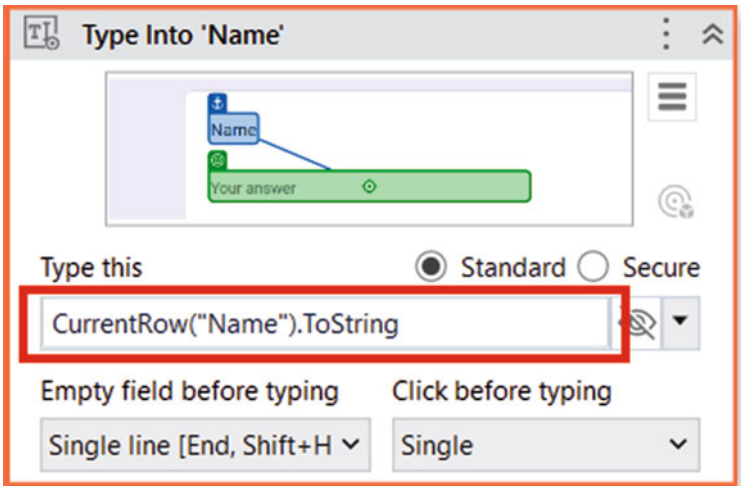



**Fig. 13.19** Step 17: Assign values from the csv file to variables

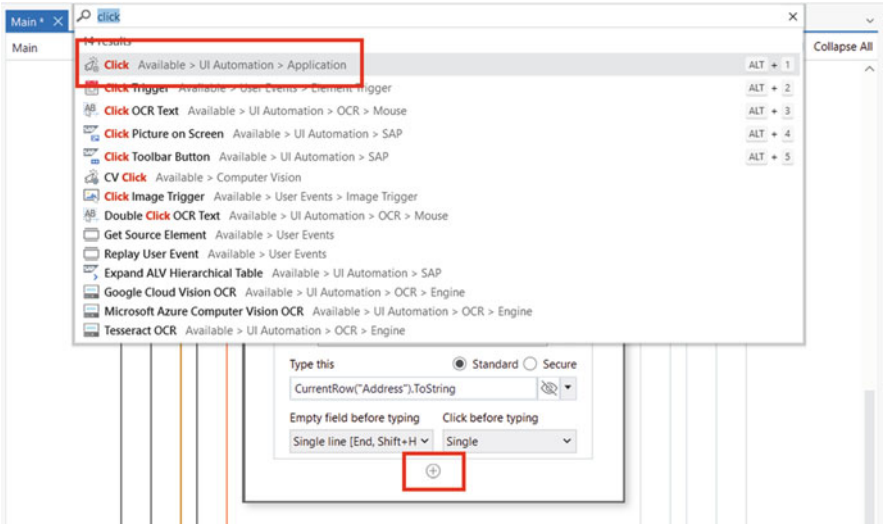



**Fig. 13.20** Step 18: Add click activity

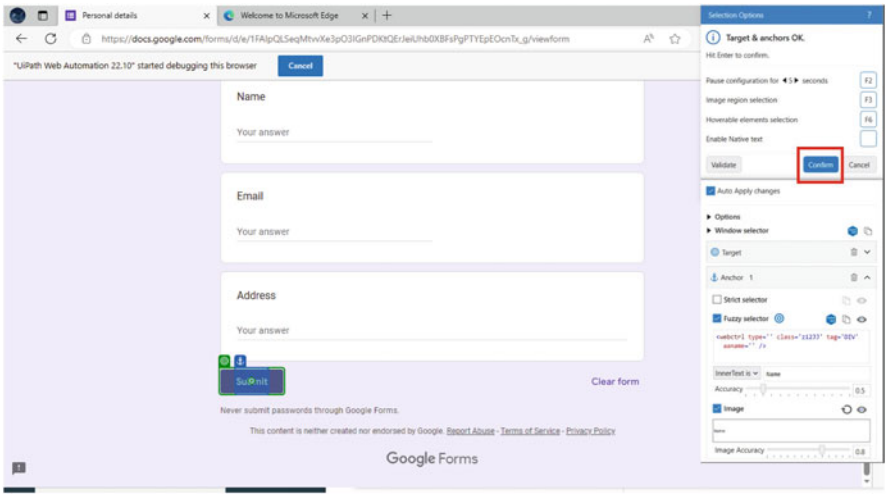



**Fig. 13.21** Step 19: Anchor submit button

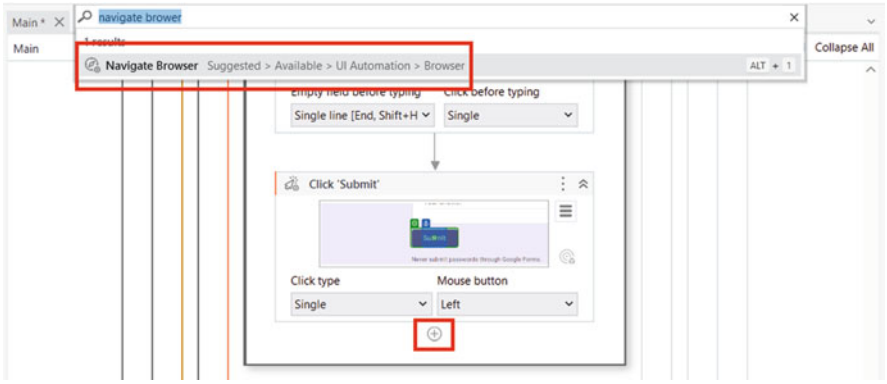
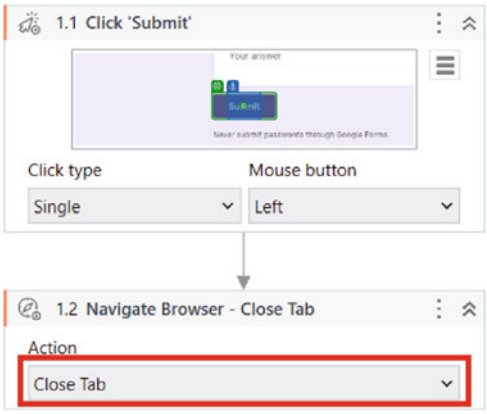



**Fig. 13.22** Step 20: Add navigate browser activity

**Fig. 13.23** Step 21: Automate tab closing



## Exercises

1. List three different roles and responsibilities of operations departments.
2. Explain the purpose of the business process management and list the five steps involved.
3. Explain the purpose of Six Sigma and list the five steps involved.
4. Identify three different challenges in operations management.
5. For each of the challenges listed in 4, identify one way that artificial intelligence could potentially be used to alleviate the problem.
6. Develop a process automation robot that can help create new accounts and send confirmation emails using UiPath.

## References

1. Aryashah2k (2022) Incident Root Cause Analysis. https://www.kaggle.com/code/aryashah2k/incident-root-cause-analysis. Accessed 27 Apr 2023
2. Gary (2021) What is Operations Management - Definition & Guide. https://checkflow.io/what-is-operations-management. Accessed 27 Apr 2023
3. Ramswarup K (2020) Automating a Google Form Filling Task Using UiPath - GeeksforGeeks. https://www.geeksforgeeks.org/automating-a-google-form-filling-task-using-uipath/. Accessed 27 Apr 2023
4. Sakshi G (2022) Supply Chain Management vs Operations Management. https://www.tutorialspoint.com/supply-chain-management-vs-operations-management. Accessed 27 Apr 2023
5. VectorMine (2023) BPM Vector Illustration Outlined Business Process Stock Vector (Royalty Free) 1341921746 | Shutterstock. https://www.shutterstock.com/image-vector/bpm-vector-illustration-outlined-business-process-1341921746. accessed 13 May 2023
6. VectorMine (2023) Six Sigma Techniques Tools Process Improvement Stock Vector (Royalty Free) 1999676261 | Shutterstock. https://www.shutterstock.com/image-vector/six-sigma-techniques-tools-process-improvement-1999676261. Accessed 13 May 2023
7. UCIML (2016) Air pressure system failures in Scania trucks. https://www.kaggle.com/datasets/uciml/aps-failure-at-scania-trucks-data-set. Accessed 27 Apr 2023

# Chapter 14
# AI in Corporate Finance


**Learning outcomes:**

- Understand the roles and responsibilities of corporate finance departments.
- Be able to list some of the challenges that corporate finance departments face.
- Identify ways that artificial intelligence can be applied to corporate finance.
- Develop artificial intelligence solutions for predicting credit defaults and predicting credit card fraud.

## 14.1 Introduction to Corporate Finance

The finance department is in charge of overseeing all operations within the company that utilizes money. Tasks such as financing, risk analysis, investments, accounting, and compliance are performed by the finance department [1].

**Financing**
A firm needs money, or capital, to operate, and there are multiple ways for them to get it. The finance department is in charge of deciding the capital structure of the business and sourcing for the capital to fund operations. There are many instruments that a business can use to finance their operations. These include corporate bonds, bank loans, equities, and other financial instruments. In order to secure the capital required, the finance department negotiates loan terms, connects with potential investors, and maintains relations with the existing shareholders. They also have to consider the pros and cons of each financing option to best match the needs of the company.

**Investments**
Business leaders seek guidance from the finance department about how to invest available funds. Advisors in the finance department identify lucrative opportunities



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_14

that business can take. They compare different investment projects (both internal and external to the company) by analyzing the projected returns of various opportunities as well as the capital required to fund these projects. They will also have to decide which projects to fund and how much to fund these projects in a way that would help maximize profits for the company.

**Risk Analysis**
Assessing financial risks is one of the ways the financial department supports the success of a business. They can manage risk by reviewing the terms of loans, researching the credit history of their clients, and reviewing price changes in their industry. By conducting financial analysis on the operations of the business, the financial department can advise the business on the best way forward to minimize financial risk.

**Account Management**
Accounts receivable and accounts payable are overseen by members of the finance department. They make ensure that the organization fulfills its contractual commitments by paying the money that it owes, sending out invoices, and making steps to reclaim money that is still owed to it. Finance departments are also in charge of tracking and documenting all transactions to facilitate financial audits.

**Compliance**
The finance department also contributes to the maintenance of compliance with applicable financial legislation and policies. This involves conducting an audit of the firm's financial procedures, engaging with regulatory officials, and making certain that corporate assets and financial records are in agreement with one another. Additionally, they are responsible for the preparation of any necessary reports for the public disclosure of the company's earnings or financial policies.

## 14.2 Artificial Intelligence in Finance

The use of artificial intelligence to corporate finance is of the utmost significance. Corporate finance investment and transactions are key forms of corporate activity that have a considerable influence not just on specific industrial sectors but also on national and regional economies and, by extension, on society as a whole.

(1) **Process automation**
Process automation can be used in the finance department to streamline repetitive processes. Processing claims by extracting information and verifying the legitimacy of the transactions can be performed using artificial intelligence and robotic process automation to detect and recognize texts in an invoice. Documents can also be classified so that they can be properly tagged and routed to the right teams. Such examples can help reduce time spent by finance teams and allow them to perform more complex tasks.

(2) **Credit defaults**
Bankruptcy and defaults are significant risks to the company as they may never be able to receive the cash for services provided or even capital that was loaned to debtors. We can utilize AI to predict bankruptcy, taking into account the financial status of the companies that we are transacting with. A higher accuracy in predicting defaults would allow business to better anticipate and plan to collect money back effectively.

(3) **Fraud detection**
Fraud is another financial risk that companies face. Businesses can lose money or inventory through fraudulent transactions that are performed by bad actors. Chargeback fraud is one example, where customers request a refund for a transaction directly with the payment processor after receiving the goods, through the chargeback process without informing the business. Such cases lead to a loss of inventory as the costs would be borne by the business. Artificial Intelligence can be utilized to reduce such risks by flagging out potentially fraudulent activity and provide a probability that a transaction is fraudulent.

(4) **Enforce regulation**
Business operations are required to comply with financial regulations in the country of operation, and failure to comply with these regulations can lead to detrimental consequences for a business. One such example is anti-money laundering laws, which are created to prevent illicit funds from being disguised as legitimate transactions. In such cases, artificial intelligence may be used to help comply with regulation requirements through facilitating identity verification and conducting customer due diligence.

## 14.3 Applications of AI in Corporate Finance

### *14.3.1 Default Prediction*

Credit risk modeling is the process of assessing the likelihood of a borrower defaulting on their loan [2]. This is accomplished by extracting insights from historical data of previous borrowers. Banks and other financial organizations utilize credit risk models in order to improve their decision-making processes about the individuals to whom they lend money, the amount of money they lend, and when they should draw back. Similarly, businesses can apply the same techniques to determine the credit risk of their customers' business to business (B2B) credit transactions.

When businesses have a better grasp of the elements that contribute to the risk that a borrower would default on their loans, they are able to make more educated decisions on who to lend money to, how much money to extend, and when to draw back. One way they do this is by improving their ability to predict credit risk by using machine learning. Machine learning models are able to draw insights from

past data and identify customers which may default on their loans. In addition, algorithms that use machine learning may acquire new knowledge from data over time, which allows them to improve their predictions as more data are collected.

In this exercise we will be predicting whether a company will or will not become bankrupt and hence default on their payments. We will achieve this using Support Vector Machines with SMOTE.

First, let us begin by downloading and importing the data. The dataset we will be using can be obtained from [3]:
https://www.kaggle.com/datasets/lupitaastari/bankruptcy?select=bankruptcy_Train.csv.

We will split the data provided in bankruptcy_Train into training and testing as the test set provided does not provide labels for us to verify how successful our model would be in predicting bankruptcy.

```
import pandas as pd
import numpy as np
df = pd.read_csv("Finance/bankruptcy_Train.csv")
fields = pd.read_excel("Finance/bankruptcy DataField.
↪xlsx")
```

```
df
```

```
         Attr1     Attr2     Attr3     Attr4     Attr5     Attr6     Attr7  \
0    -0.031545 -0.091313 -0.040269 -0.013529  0.007406 -0.016047 -0.000264
1    -0.231729 -0.049448  0.304381 -0.080975  0.007515 -0.016047 -0.034963
2    -0.058602  0.065060 -0.488404 -0.189489  0.006572 -0.016047 -0.004954
3    -0.069376  0.044641 -0.181684 -0.140032  0.007477 -0.010915 -0.005599
4     0.236424 -0.051912  0.678337 -0.014680  0.007879 -0.016047  0.057418
...        ...       ...       ...       ...       ...       ...       ...
9995 -0.079533  0.034814 -0.492082 -0.189873  0.006687 -0.006462 -0.008582
9996 -0.081046 -0.095260  0.184167  0.021280  0.007497 -0.034968 -0.009689
9997 -0.230571  0.061341 -0.830634 -0.222373  0.006716 -0.013742 -0.042210
9998 -0.108156  0.029524  0.102420 -0.042692  0.008123 -0.018374 -0.013544
9999 -0.068674 -0.081793  0.734155  0.039538  0.007850  0.001952 -0.005791

         Attr8     Attr9    Attr10  ...    Attr56    Attr57    Attr58  \
0     0.641242 -0.748385  0.126789  ...  0.014367  0.005457 -0.014143
1     0.074710  0.469815  0.073759  ...  0.008492 -0.008385 -0.008666
2    -0.456287  0.270351 -0.071287  ...  0.010819  0.006779 -0.009437
3    -0.462971 -0.286746 -0.085266  ...  0.010683  0.005384 -0.010840
4     0.097183  0.423405  0.076880  ...  0.010970  0.025295 -0.011056
...        ...       ...       ...  ...       ...       ...       ...
9995 -0.374739 -0.372026 -0.034110  ...  0.009554  0.003007 -0.009712
9996  0.689695 -0.393121  0.123218  ...  0.009243  0.002485 -0.009400
9997 -0.471830 -0.351828 -0.084068  ...  0.009841 -0.025046 -0.009998
9998 -0.355796 -0.480887 -0.026274  ...  0.009595 -0.000439 -0.009101
9999  0.293253 -0.398417  0.066852  ...  0.009162  0.003478 -0.009319

        Attr59    Attr60    Attr61    Attr62    Attr63    Attr64  class
0    -0.020924  0.068399 -0.214478 -0.013915 -0.173939 -0.046788      0
1    -0.023095 -0.033498 -0.205796 -0.015174 -0.073056 -0.027236      0
2    -0.007919 -0.043455  0.019740 -0.011736 -0.291624 -0.033580      0
3     0.001381 -0.042828 -0.350519  0.002969 -0.554685 -0.046823      0
4    -0.022535 -0.035892 -0.181557 -0.015623 -0.027841 -0.023694      0
```

(continues on next page)

(continued from previous page)

```
...        ...       ...       ...       ...       ...       ...    ...
9995 -0.016201 -0.043479 -0.320006  0.002964 -0.554635 -0.043685      0
9996 -0.021760 -0.040991  0.081487 -0.017605  0.275468 -0.040571      0
9997 -0.009408 -0.038734 -0.250595 -0.005232 -0.463932 -0.040395      0
9998 -0.005170 -0.022103 -0.298745 -0.013596 -0.194810 -0.043671      0
9999 -0.023124 -0.015060 -0.029375 -0.018460  0.503620 -0.017615      0

[10000 rows x 65 columns]
```

```
{i.split("\xa0- ")[0]:i.split("\xa0- ")[1] for i in␣
↪fields["Data fields"].tolist()}
```

```
{'attr1': 'net profit / total assets',
 'attr2': 'total liabilities / total assets',
 'attr3': 'working capital / total assets',
 'attr4': 'current assets / short-term liabilities',
 'attr5': '[(cash + short-term securities + receivables - short-term liabilities) /
↪ (operating expenses - depreciation)] * 365',
 'attr6': 'retained earnings / total assets',
 'attr7': 'EBIT / total assets',
 'attr8': 'book value of equity / total liabilities',
 'attr9': 'sales / total assets',
 'attr10': 'equity / total assets',
 'attr11': '(gross profit + extraordinary items + financial expenses) / total␣
↪assets',
 'attr12': 'gross profit / short-term liabilities',
 'attr13': '(gross profit + depreciation) / sales',
 'attr14': '(gross profit + interest) / total assets',
 'attr15': '(total liabilities * 365) / (gross profit + depreciation)',
 'attr16': '(gross profit + depreciation) / total liabilities',
 'attr17': 'total assets / total liabilities',
 'attr18': 'gross profit / total assets',
 'attr19': 'gross profit / sales',
 'attr20': '(inventory * 365) / sales',
 'attr21': 'sales (n) / sales (n-1)',
 'attr22': 'profit on operating activities / total assets',
 'attr23': 'net profit / sales',
 'attr24': 'gross profit (in 3 years) / total assets',
 'attr25': '(equity - share capital) / total assets',
 'attr26': '(net profit + depreciation) / total liabilities',
 'attr27': 'profit on operating activities / financial expenses',
 'attr28': 'working capital / fixed assets',
 'attr29': 'logarithm of total assets',
 'attr30': '(total liabilities - cash) / sales',
 'attr31': '(gross profit + interest) / sales',
 'attr32': '(current liabilities * 365) / cost of products sold',
 'attr33': 'operating expenses / short-term liabilities',
 'attr34': 'operating expenses / total liabilities',
 'attr35': 'profit on sales / total assets',
 'attr36': 'total sales / total assets',
 'attr37': '(current assets - inventories) / long-term liabilities',
 'attr38': 'constant capital / total assets',
 'attr39': 'profit on sales / sales',
 'attr40': '(current assets - inventory - receivables) / short-term liabilities',
 'attr41': 'total liabilities / ((profit on operating activities + depreciation) *␣
↪(12/365))',
 'attr42': 'profit on operating activities / sales',
 'attr43': 'rotation receivables + inventory turnover in days',
 'attr44': '(receivables * 365) / sales',
 'attr45': 'net profit / inventory',
 'attr46': '(current assets - inventory) / short-term liabilities',
 'attr47': '(inventory * 365) / cost of products sold',
```

(continues on next page)

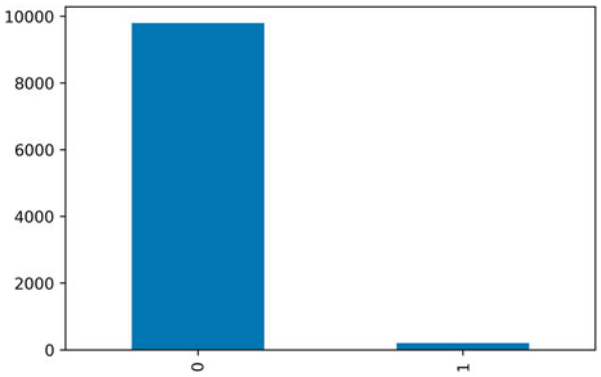
(continued from previous page)

```
 'attr48': 'EBITDA (profit on operating activities - depreciation) / total assets',
 'attr49': 'EBITDA (profit on operating activities - depreciation) / sales',
 'attr50': 'current assets / total liabilities',
 'attr51': 'short-term liabilities / total assets',
 'attr52': '(short-term liabilities * 365) / cost of products sold)',
 'attr53': 'equity / fixed assets',
 'attr54': 'constant capital / fixed assets',
 'attr55': 'working capital',
 'attr56': '(sales - cost of products sold) / sales',
 'attr57': '(current assets - inventory - short-term liabilities) / (sales - gross␣
↪profit - depreciation)',
 'attr58': 'total costs /total sales',
 'attr59': 'long-term liabilities / equity',
 'attr60': 'sales / inventory',
 'attr61': 'sales / receivables',
 'attr62': '(short-term liabilities *365) / sales',
 'attr63': 'sales / short-term liabilities',
 'attr64': 'sales / fixed assets',
 'class': 'the response variable Y: 0 = did not bankrupt; 1 = bankrupt'}
```

Then let us visualize the class counts.

```
pd.value_counts(df["class"]).plot.bar()
```

```
<AxesSubplot:>
```



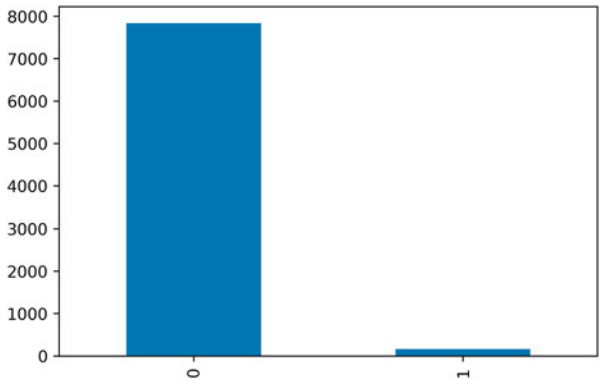

Split the data into training and testing.

```
train = df[:int(0.8*(len(df)))]
test = df[int(0.8*(len(df))):]
```

Visualize the class counts for train and test, respectively.

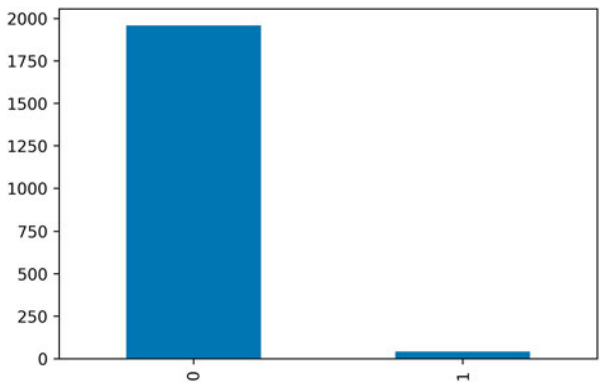
```
pd.value_counts(train["class"]).plot.bar()
```

```
<AxesSubplot:>
```




```
pd.value_counts(test["class"]).plot.bar()
```

```
<AxesSubplot:>
```




```
pd.value_counts(df["class"])[1]/ pd.value_counts(df[
↪"class"])[0]
```

```
0.020720628763907317
```

Let us split the data into features (X) and labels (Y).

```
from collections import Counter
train_X, train_Y = train.drop("class", axis = 1),␣
↪train["class"]
```

(continues on next page)

(continued from previous page)

```
test_X, test_Y = test.drop("class", axis = 1), test[
↪"class"]
# summarize the class distribution
counter = Counter(train_Y)
print(counter)
```

```
Counter({0: 7838, 1: 162})
```

As you can see, the class distribution is highly imbalanced. Bond defaults only account for approximately 2% of the dataset. Training a classifier on such a dataset would yield bad results as the model would simply predict all transactions as unlikely to default since that would lead to a 98% accuracy.

For example, let us try a Logistic Regression classifier with the current imbalanced data.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import balanced_accuracy_score,␣
↪recall_score, confusion_matrix

model = LogisticRegression(max_iter = 1000)
model.fit(train_X, train_Y)
prediction = model.predict(test_X)

confusion_matrix(test_Y, prediction)
```

```
array([[1956,    3],
       [  41,    0]], dtype=int64)
```

```
balanced_accuracy_score(test_Y, prediction)
```

```
0.4992343032159265
```

```
recall_score(test_Y, prediction, average = None)
```

```
array([0.99846861, 0.        ])
```

As you can see from the confusion matrix and the recall scores, due to the class imbalance, the model predicts that no one will become bankrupt as such cases are very rare. In order to combat the imbalanced dataset, we can use SMOTE for generating more of the minority class.

SMOTE operates by selecting 2 samples that are close in the feature space, calculating a line between them in the feature space, and generating a new point by sampling a new point along the line.

Specifically, a random sample from the minority class is selected to begin with. Then we searched for k of its nearest neighbors for that sample. One of these selected neighbors is chosen and a new example is created randomly at a selected point that lies between the two examples in feature space.

Let us see how it works.

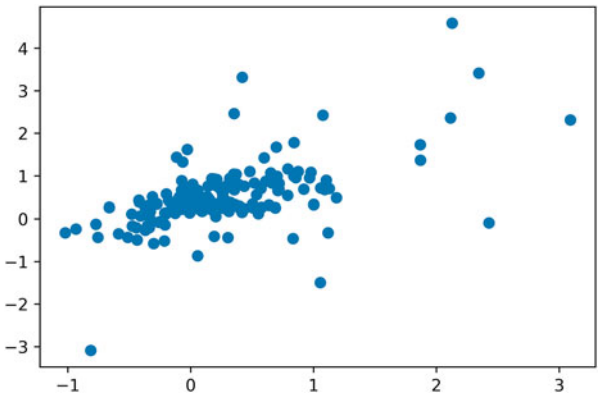
```
from imblearn.over_sampling import SMOTE
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(train_X)
pca_train = pca.transform(train_X)
```

This is a plot of the minority class before SMOTE.

```
import matplotlib.pyplot as plt
plt.scatter(pca_train[train_Y==1][:,0], pca_
↪train[train_Y==1][:,1])
```

```
<matplotlib.collections.PathCollection at␣
↪0x1b64eeabd30>
```




```
oversample = SMOTE(random_state=0)
train_X, train_Y = oversample.fit_resample(train_X,␣
↪train_Y)
```

(continues on next page)

(continued from previous page)

```
# summarize the new class distribution
counter = Counter(train_Y)
print(counter)
```

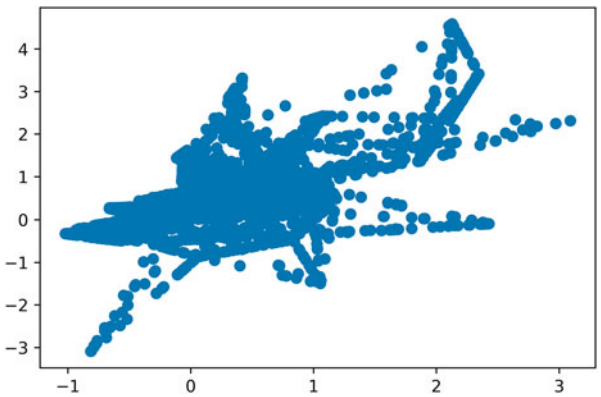
```
Counter({0: 7838, 1: 7838})
```

```
pca_train = pca.transform(train_X)
```

This is a plot of the minority sample after SMOTE.

```
plt.scatter(pca_train[train_Y==1][:,0], pca_
↪train[train_Y==1][:,1])
```

```
<matplotlib.collections.PathCollection at␣
↪0x1b64fd1bdc0>
```




Now that we have equal classes in the training set, let us train the Logistic Regression classifier once more.

```
model = LogisticRegression(max_iter = 1000, random_
↪state=0)
model.fit(train_X, train_Y)
prediction = model.predict(test_X)
```

```
confusion_matrix(test_Y, prediction)
```

```
array([[1478,  481],
       [   5,   36]], dtype=int64)
```

```
balanced_accuracy_score(test_Y, prediction)
```

```
0.8162576725307835
```

```
recall_score(test_Y, prediction, average = None)
```

```
array([0.75446656, 0.87804878])
```

As you can see, we have achieved 80.4% in the balanced accuracy score, and we were able to recall 85% of all defaults (and hence, avoid purchasing) in the test set.

Using Logistic Regression, we are able to predict the probability of default. The earlier model predicts a default when the probability of default is greater than 50%. We can lower it to be more conservative to recall more defaults, but some bonds that we have previously deemed unlikely to default may now also be considered to be at risk of defaulting.

```
probability_of_default = model.predict_proba(test_
↪X)[:,1]
```

```
threshold_confidence = 0.2
prediction = (probability_of_default > threshold_
↪confidence).astype(np.int)
```

```
confusion_matrix(test_Y, prediction)
```

```
array([[1144,  815],
       [   1,   40]], dtype=int64)
```

```
balanced_accuracy_score(test_Y, prediction)
```

```
0.7797905850421445
```

```
recall_score(test_Y, prediction, average = None)
```

```
array([0.58397141, 0.97560976])
```

As you can see, setting a lower threshold of 20% instead of 50% on our model allows us to be more conservative. As a result, we detected 97.6% of all defaults.

### 14.3.2 Predicting Credit Card Fraud

Chargeback fraud is when a person dishonestly disputes a payment transaction through the issuing bank or the payment processor, which ultimately results in a chargeback [4]. The fundamental objective of the purchaser is to acquire a refund while retaining possession of the good(s). The customer starts a chargeback on purpose instead of contacting the business where they made the purchase. Through this process, they are able to steal from the company through the chargeback process.

Every chargeback results in a cost being imposed to the business. If a business wants to dispute a chargeback, they usually go through a long dispute process. In the event that the consumer wins their case throughout the dispute procedure, the company is obligated to issue a refund for the amount that was originally charged to the client. Businesses lose billions of dollars yearly due to chargebacks, and a rapidly growing percentage of that loss is attributed to fraudulent chargebacks.

It is essential for businesses to maintain a low chargeback rate in order to avoid incurring excessive costs and maintain their ability to accept major credit cards. With the prevalence of both chargeback fraud and friendly fraud on the rise, it is becoming increasingly hard for businesses to keep their chargeback rate low.

In this example we will develop a model that can accurately detect fraudulent credit card transactions [5]. The problem is significant because false positive fraud detections can result in inconvenience for customers, while false negatives can result in monetary losses for the credit card company.

The dataset contains transactions made by European credit cardholders in September 2013. The transactions have been transformed with Principal Component Analysis (PCA), with the exception of the "Time" and "Amount" features. The "Class" feature represents whether a transaction is fraudulent (1) or not (0). The dataset is highly imbalanced, with only 0.172% of transactions being fraudulent.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from imblearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix,␣
↪balanced_accuracy_score, accuracy_score, recall_
↪score
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE
```

```
df=pd.read_csv("Finance/creditcard.csv")
```

```
# Shape of the dataset
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (284807, 31)
```

```
# Overview of the data
print("Overview of the data:" )
df.head()
```

```
Overview of the data:
```

```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
```

(continues on next page)

(continued from previous page)

```
 8   V8       284807 non-null  float64
 9   V9       284807 non-null  float64
 10  V10      284807 non-null  float64
 11  V11      284807 non-null  float64
 12  V12      284807 non-null  float64
 13  V13      284807 non-null  float64
 14  V14      284807 non-null  float64
 15  V15      284807 non-null  float64
 16  V16      284807 non-null  float64
 17  V17      284807 non-null  float64
 18  V18      284807 non-null  float64
 19  V19      284807 non-null  float64
 20  V20      284807 non-null  float64
 21  V21      284807 non-null  float64
 22  V22      284807 non-null  float64
 23  V23      284807 non-null  float64
 24  V24      284807 non-null  float64
 25  V25      284807 non-null  float64
 26  V26      284807 non-null  float64
 27  V27      284807 non-null  float64
 28  V28      284807 non-null  float64
 29  Amount   284807 non-null  float64
 30  Class    284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
df.describe()
```

```
                Time             V1             V2             V3             V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5             V6             V7             V8             V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
```

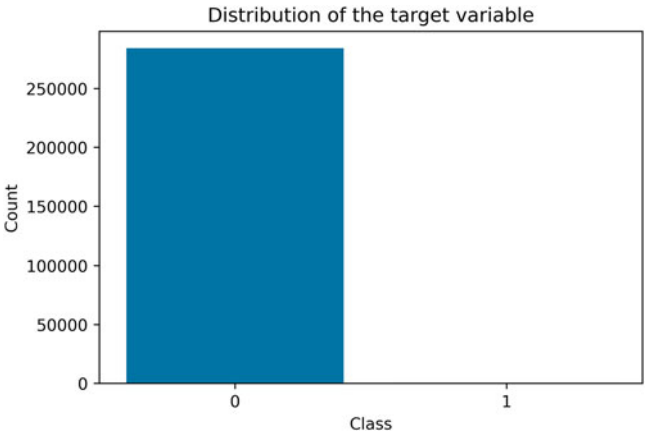
(continues on next page)

(continued from previous page)

```
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

We will take a look and count how many transactions are not fraudulent and compare it against how many that are fraudulent.

```
#Plotting the count of each class in the target␣
↪variable
sns.countplot(x='Class', data=df)

#Adding the title, x-axis label and y-axis label to␣
↪the plot
plt.title("Distribution of the target variable")
plt.xlabel("Class")
plt.ylabel("Count")

#Displaying the plot
plt.show()
class_counts = df['Class'].value_counts()
print(class_counts)
```




```
0    284315
1       492
Name: Class, dtype: int64
```

As shown by the plots, the dataset is highly imbalanced, with the bulk samples belonging to class 0 (not fraudulent) and only a tiny percentage belonging to class 1 (fraudulent) (fraudulent). Specifically, there are 284315 examples from class 0 and just 492 samples from class 1. This imbalance in the target variable will hurt the performance of the machine learning models because it is likely that the model will be biased toward the majority class.

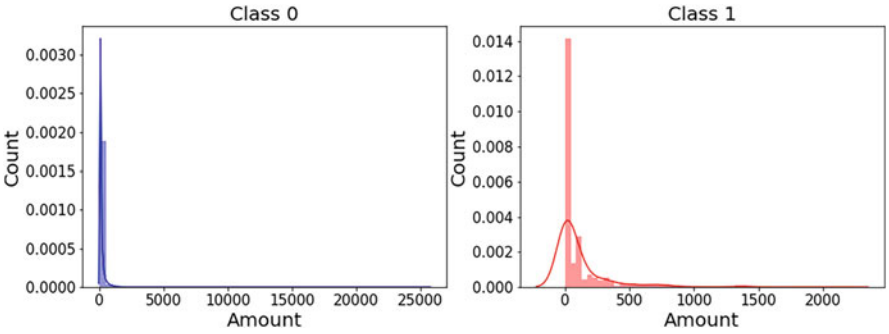
Now let us plot the distribution of the amount transacted for both non-fraudulent and fraudulent transactions. The first plot shows the distribution of class 0 (non-fraud) in blue with respect to the amount transacted. On the other hand, the second plot shows the distribution for class 1 (fraud) in red. By comparing the two groups visually, we can identify notable differences that help separate the fraudulent transactions from the non-fraudulent ones.

```
# Create a subplot with two plots side by side
fig, axs = plt.subplots(1, 2, figsize=(15, 5))

# Plot distribution of amount for class 0
sns.distplot(df[df['Class'] == 0]['Amount'], kde=True,
↪ ax=axs[0], color='b')
axs[0].set_title('Class 0')
axs[0].set_xlabel('Amount')
axs[0].set_ylabel('Count')

# Plot distribution of amount for class 1
sns.distplot(df[df['Class'] == 1]['Amount'], kde=True,
↪ ax=axs[1], color='r')
```

(continues on next page)

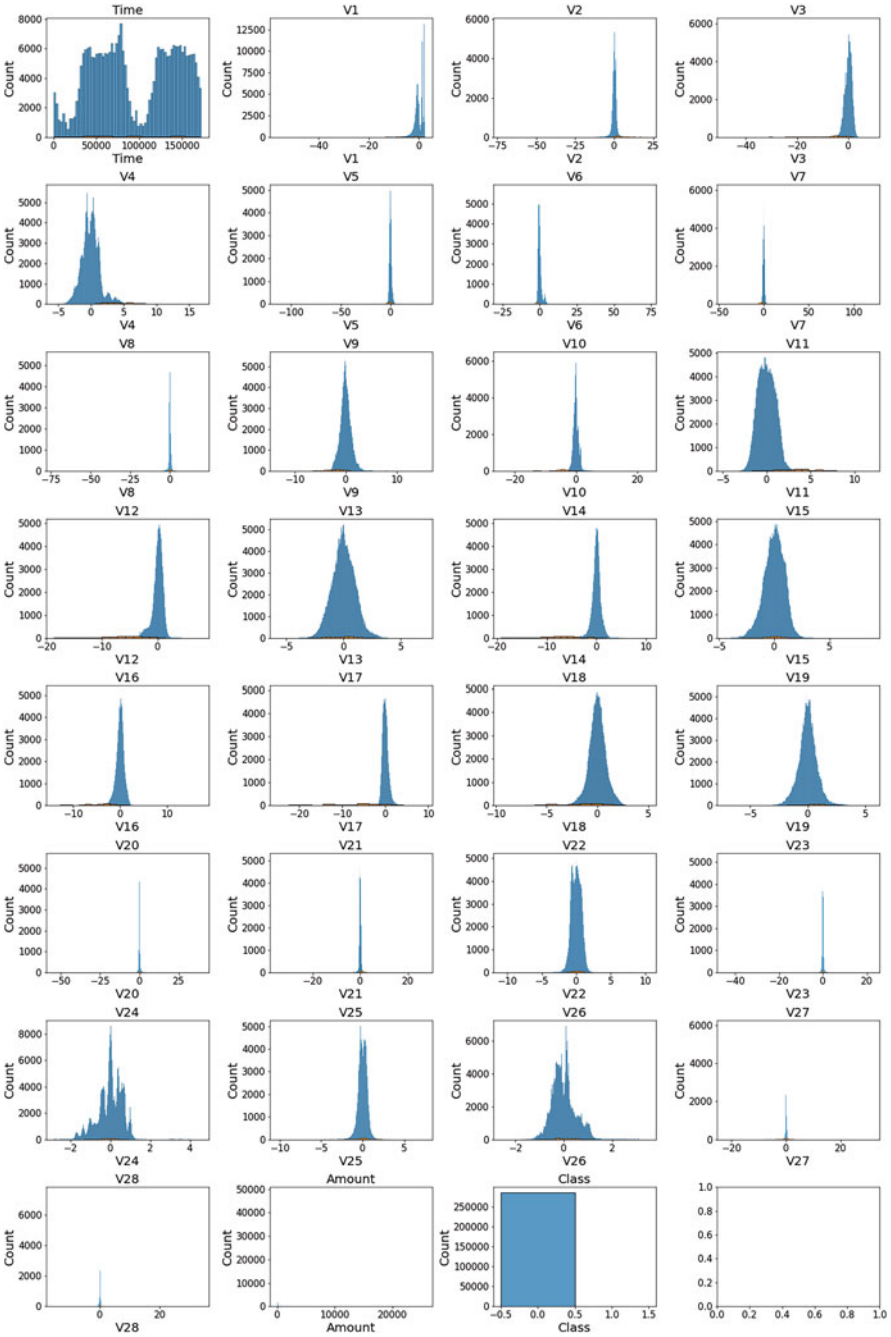
(continued from previous page)

```
axs[1].set_title('Class 1')
axs[1].set_xlabel('Amount')
axs[1].set_ylabel('Count')

plt.show()
```




As you can see, the red plot has a smaller range of amounts transacted than the blue plots, indicating that fraudulent transactions usually happen with small amounts transacted to avoid drawing attention.

```
# Create a subplot with multiple plots
fig, axs = plt.subplots(8, 4, figsize=(20, 30))
axs = axs.ravel()

# Plot histograms for all columns
for i, column in enumerate(df.columns):
    sns.histplot(df[df['Class'] == 0][column], ␣
↪ax=axs[i])
    sns.histplot(df[df['Class'] == 1][column], ␣
↪ax=axs[i], color='tab:orange')
    axs[i].set_title(column)

plt.tight_layout()
plt.show()
```




Let us visualize the histograms of all features in our dataset so that we can better understand the distribution of each of the variables. Examining the histograms

enables us to determine if the data is regularly distributed, skewed to the left or right, or has several peaks. This information can help with preprocessing the data, like transforming or scaling the variables, and choosing the right modeling techniques.

```
# Split the data into features and target
X = df.drop('Class', axis=1)
y = df['Class']
```

Sampling and data scaling are crucial steps in the data preparation process. However, it is important to perform these operations only using the training data to avoid data leakage. Data leakage occurs when information from the test set is used to train the model, which can result in overfitting and poor performance on new unseen data.

```
X_train, X_test, y_train, y_test = train_test_split(X,
 ↪ y, test_size=0.3, random_state=0)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

sampler = SMOTE(random_state=0)
X_train, y_train = sampler.fit_resample(X_train, y_
 ↪train)

model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Now that we have generated our predictions, let us evaluate the results.

```
# Calculate precision, recall, and f1-score
accuracy = accuracy_score(y_test, y_pred)
balanced_accuracy = balanced_accuracy_score(y_test, y_
 ↪pred)
recall_fraud = recall_score(y_test, y_pred)
recall_non_fraud = recall_score(y_test, y_pred, pos_
 ↪label=0)
print(recall_non_fraud)


# Print the metrics
print('Accuracy Score: {:.3f}'.format(accuracy.
 ↪mean()))
print('Recall Fraudulent: {:.3f}'.format(recall_fraud.
 ↪mean()))
```

(continues on next page)

(continued from previous page)

```
print('Recall Non-Fraudulent: {:.3f}'.format(recall_
↪non_fraud.mean()))
print('Balanced Accuracy Score: {:.3f}'.
↪format(balanced_accuracy.mean()))
print(confusion_matrix(y_test, y_pred))
```

```
0.9760715625586194
Accuracy Score: 0.976
Recall Fraudulent: 0.918
Recall Non-Fraudulent: 0.976
Balanced Accuracy Score: 0.947
[[83255  2041]
 [   12   135]]
```

As you can see, we have successfully recalled 91.8% of all fraudulent transactions while ensuring that 97.6% of normal transactions would not be flagged out by our model. Only 2.4% of transactions may be declined or require further verification.

## Exercises

(1) List three different roles and responsibilities of corporate finance departments.
(2) Be able to list some of the challenges that corporate finance departments face.
(3) Identify three different challenges in corporate finance.
(4) For each of the challenges listed in 3, identify one way that artificial intelligence could potentially be used to alleviate the problem.
(5) Develop artificial intelligence solutions for predicting e-commerce fraud.
We will utilize this dataset [6]:
https://www.kaggle.com/datasets/vbinh002/fraud-ecommerce.
This dataset consists of information collected about various purchase transactions.
In this exercise, try to use what you have learnt so far to develop a model that can classify whether a transaction is fraudulent or not.

## References

1. Team IE (2023) What is the role of the finance department in business? In: Indeed. https://sg.indeed.com/career-advice/career-development/role-of-finance-department. Accessed 27 Apr 2023
2. Ajitesh K (2022) Credit risk modeling and machine learning use cases—data analytics. https://vitalflux.com/credit-risk-modeling-machine-learning-use-cases/. Accessed 27 Apr 2023

3. Lupitaastari (2020) Bankruptcy dataset. In: Kaggle. https://www.kaggle.com/datasets/lupitaastari/bankruptcy. Accessed 27 Apr 2023
4. Team F (2022) Chargeback fraud: what it is and how to prevent it. https://www.forter.com/blog/chargeback-fraud-what-it-is-and-how-to-prevent-it/. Accessed 27 Apr 2023
5. Mlg-ulb (2019) Credit card fraud detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. [Online; accessed 2023-02-13], Accessed 27 Apr 2023
6. Vu B (2018) Fraud ecommerce. https://www.kaggle.com/datasets/vbinh002/fraud-ecommerce

# Chapter 15
# AI in Business Law


**Learning outcomes:**

- Recognize what constitutes business law.
- Analyze the differences between business structures and remember the considerations when selecting one to use.
- Be able to list the different types of business laws.
- Identify how artificial intelligence can be applied in business laws.
- Develop artificial intelligence solutions for legal document summarization, contract review, and legal research.

## 15.1 Introduction to Business Law

A law is a widely accepted and upheld concept or causal connection that, when broken, carries a punishment like harm, suffering, failure, or loss [12]. Laws are enforceable regulations that govern how people and corporations must behave. They are created to establish responsibility and uphold the rule of law. A law is primarily the result of formal or informal legislation by a local authority figure, such as a legislature or a king. Laws have the authority and power of the enacting entity, and anyone who disobeys them faces repercussions.

Laws get their legitimacy by being founded on commonly recognized ideas, such as the sovereign authority of the person passing them or the justice of each law. Laws, which are frequently derived through observations or experiments, define precise connections between the causes and effects of occurrences.

Business law, often known as mercantile law or commercial law, is the body of regulations that governs businesses engaged in commercial activities [5]. Business law covers all laws that are relevant to businesses which include formation law,



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_15

employment law, intellectual property law, contract law and negotiations, taxes, lawsuits, bankruptcy laws, and antitrust laws. These regulations may be passed by:

- Agreement
- Convention
- International or national legislation

Businesses of all types have one thing in common: they must follow the required guidelines and regulations [5]. Whether it's formation laws that assist businesses in getting started, or bankruptcy laws that help a business dissolve, the entire lifeline of a business is built on these rules and norms that regulate it and give it structure.

There are various types of business structures and each of which is governed differently by the law. Below are some of the considerations in selecting a suitable business structure [3]:

- **Taxes**
  Tax obligations, payers, and applicable tax exemptions vary based on how a business is organized.
- **Liabilities**
  In certain business structures, the owner's personal assets may be at risk if the business fails to meet its financial obligations.
- **Hierarchies**
  The size and responsibilities of corporate ladders vary based on the business formation structure chosen.
- **Documents and Permits**
  The process of filing with entities and obtaining necessary permits will differ depending on the type of business structure. As a result, the annual reporting and documentation requirements will differ.

### *15.1.1 Types of Businesses*

Selecting a suitable type of business to operate is an important strategy that will affect how your business is run. Different types of business structures have different responsibilities, liabilities, privileges, and restrictions. Below are some of the different types of business structures:

- **Sole Proprietorship**
  In a sole proprietorship, you and your businesses are the same. The advantage is that the formation costs are lower, and you have complete control over all business decisions. The disadvantage is that you have no liability protection and are putting your personal assets at risk.
- **Partnership**
  A partnership is formed when several people share their resources to run a business. This can be done in a variety of ways, but it is usually formalized

through a partnership agreement that specifies partner ownership and profit/loss percentages within the business.

- General Partnership: All partners in a general partnership have equal rights and obligations. The day-to-day management of the company is split among partners.
- Limited Partnership: Some partners in this business structure are "silent partners," meaning they are not held liable for the company's responsibilities. Their main obligation is to invest money. In limited partnerships, at least one general partner is in charge of making decisions. The personal assets of the silent partners are insulated from responsibility due to their minimal involvement in the business. The general partners are the ones putting their own money at risk.
- Limited Liability Partnership: Every partner is accountable for making business decisions. Each partner, however, is insulated from the liabilities of the other partners. Partners are protected from the possibility of losing personal assets if one of the other partners is sued.
- Limited Liability Limited Partnership: These are limited partnerships where both the general and quiet partners are protected from liability.

- **Corporations**
  A corporation is a corporate entity that is owned by investors and a board of directors. The purpose of for-profit corporations is to be profitable in the best interests of the shareholders. Shareholders can benefit from the chance to return their investment while being protected from the risk of lenders pursuing their personal assets.

  - S-Corporation: S-Corp businesses are exempt from paying taxes. S-corporations can only have 100 shareholders.
  - C-Corporation: C-Corp is subject to double taxation, and thus the business will have to pay its taxes. Its shareholders, however, are liable to personal income tax. The number of members in a C-Corps is not limited.
  - B-Corporation: A B-Corp, often known as a "benefit corporation," is a certification for a corporation that ensures that the company's aim is to benefit the public (workers, customers, suppliers, etc.) rather than its shareholders. The goal is to create a profit while also benefiting society. B-Corps, like C-Corps, has a double-taxation scheme.

- **Limited Liability Companies**
  A limited liability company (LLC) is a popular business structure. The advantages of being a company include lower potential liability. LLCs prevent company members from disclosing personal assets to pay off penalties. LLCs are pass-through entities. On their personal tax returns, members record business profits and losses. This avoids the problem of double taxes that corporations incur.

  - Single Member LLC: A single-member LLC is owned by an individual or group.

  - Multi-Member LLC: A multi-member LLC has multiple persons or groups registered as business owners. Multi-member LLCs, like other businesses, must file a tax return.

- **Non-profit**
  Non-profit organizations are formed to assist various groups of individuals and public entities. Unlike other businesses, their mission statement is not purely for profit and generally includes the language of the society the firm seeks to serve. Furthermore, the charitable nature of many firms qualifies them for state and federal tax breaks.
  Non-profits and B-Corps are similar, but there are significant differences. Non-profits are not owned by anyone, whereas B-Corps are held by shareholders.
  Non-profits, on the other hand, are governed by a board of trustees. Non-profits can also host fundraisers and collect donations from investors to raise revenue. At the same time, B-Corps are limited to traditional methods such as borrowing or selling stock.

### *15.1.2 Types of Business Laws*

Business law refers to all laws that govern businesses. As such there are many different types of laws that fall under business law such as [5]:

- **Formation Law**
  Formation laws are the rules that a corporation must follow in order to be founded and recognized as a legal entity. A firm cannot legally operate until it has gotten the official recognition from the law. To ensure the legitimacy of a company's classification, the proper documentation must be filled out and the relevant criteria must be fulfilled. This is true in many areas of formation and incorporation law. Businesses must settle on a classification and their method of operation. The type of business founded will have an impact on the taxes paid, the management style, the rules to abide by at the national and state level, and much more.
- **Employment Law**
  Employment law governs both employers' and employees' rights and responsibilities. Sexual harassment, improper workplace behavior, salaries, workplace health and safety, and unlawful discrimination are all topics covered in employment law. Both recruiters and workers have certain rights and duties that must be observed according to the employment law. Employment law defines the rules that regulate these laws and also deals with scenarios and organizations that fail to follow the established employment norms.
- **Intellectual Property Law**
  Businesses constantly invest money to develop fresh, ground-breaking, and forward-thinking concepts all the time. Because there are so many cutting-edge

solutions that are always being developed, they want to be sure that their concept was not stolen or created by someone else.
Intellectual property includes inventions, intellectual and creative works, designs, trademarks, brands, and logos used in business. Intellectual property law allows businesses to protect their creative talents. Intellectual property law is divided into various sections, including copyright and trademarks.

- **Tax Law**
  Taxes are monetary charges imposed and enforced by the federal and state governments. Businesses must pay their fair share of taxes or risk suffering severe repercussions. These penalties include hefty fines and/or severe jail term. Some examples of taxes include sales tax, employee and payroll tax, income tax, and property tax.
- **Contract Law**
  Contract law governs the development, implementation, and execution of business contracts. Business contract law makes it easier for businesses and groups to make agreements. Contract law is an essential part of business law. Because many firms and enterprises are involved in numerous agreements and discussions, it is essential to find someone who can handle the responsibility of addressing the needs and desires of all parties and aiding them in making an agreement.
- **Antitrust Law**
  Antitrust laws are recommendations that aim to assist firms maintain fair competition. The purpose of antitrust legislation is to establish a level playing field for all players or enterprises in a certain industry. These regulations are in place to assist in combating businesses that amass too much power and act unfairly to the detriment of others. Market allocation, price fixing, and monopolies are some of the issues that antitrust laws attempt to address.
- **Bankruptcy Law**
  Although no one likes to file for bankruptcy, it is something that certain companies have to do. Filing for bankruptcy provides a business with various options, each with its own set of advantages and disadvantages. Part of the legal procedure for declaring bankruptcy is deciding on the best alternative to assist the struggling firm. Bankruptcy is governed by federal law. When a business decides to file for bankruptcy, it must appear in court to declare and restructure its debts.

## 15.2 Artificial Intelligence in Business Law

Laws have profound impact on every aspect of how a business operates [11]. Almost all business transactions such as sales, purchases, partnerships, mergers, acquisitions, and even reorganization are upheld by legally binding contracts. Without strong laws that help protect creative and intellectual property of businesses, innovation would not be able to flourish. Whether we realize it or not, our legal system permeates all facets of our lives and influences how we live.

In the world today, the amount of information generated for legal cases is increasing quickly. Even though there might be a lot of useful information being collected, it takes a significant amount of time to examine. This is especially the case when practicing law which can be monotonous and frustrating. At the same time, lawyers face increasing demands for speed by clients, regulatory agencies, and courts.

Even though the global market for legal services is one of the largest, it still remains incredibly under digitalized. The legal profession is famously sluggish in the uptake of new tools and technological advancements, due to its tradition-bound nature.

Artificial intelligence has a tremendous potential to improve the efficiency of legal services. Through its ability to extract information and identify patterns in large amounts of legal documents, artificial intelligence is able to find relevant information, spot errors, and identify inconsistencies. All this can help boost the productivity of lawyers and mitigate potentially expensive errors.

There are five aspects of legal activities where NLP is becoming more prevalent [4]:

- **Legal research**
  Finding information for use in legal decision-making is done through legal research. This typically entails scouring both statute and case law to discover what is relevant to the particular issue at hand. Statutes are documents produced by the legislature, whereas case laws are produced by the courts.
- **Electronic discovery**
  The process of locating and gathering electronically stored information in response to a demand for production in a legal proceeding or investigation is known as electronic discovery or e-discovery. A major challenge in this situation is sorting through the potentially hundreds of thousands of files on a standard hard drive to determine what content is relevant and what is not.
- **Contract review**
  Lawyers frequently evaluate contracts, offer suggestions for improvement, and counsel clients on whether to sign or argue for better terms. The contracts in question can range from being small and straightforward, like non-disclosure agreements (NDAs) to being quite large and intricate, numbering hundreds of pages.
  Automated contract review tools can be utilized to examine papers whose content is largely regular and predictable. The procedure entails breaking down the contract into component sections or clauses and then analyzing each one, either to extract significant information or to compare against some standard (which could be a set of other similar contracts held by a firm). Therefore, a contract review system might, for instance, flag the absence of a phrase to address bribery or highlight that a clause for price rises is missing a percentage limit.
- **Document automation**
  Document automation systems often employ a fill-in-the-blanks method that enables the development of a customized legal document based on a set of

criteria. The information needed to produce the document could, however, be acquired using an iterative question-and-answer dialog, such as a chatbot. In such situations, the document automation system's user interface is comparable to that of a legal advising system.

- **Legal advice**
  Legal advisers are interactive systems that generate advice specific to the user's needs and circumstances based on a series of questions it asks. Usually, this involves a question-and-answer format, specific to the case at hand. Legal advice could be delivered in the form of document automation since the outcome is frequently some sort of legal document.

## 15.3 Applications of AI in Business Law

### *15.3.1 Legal Document Summarization*

Legal research is a labor-intensive and time-consuming procedure. For instance, legal practitioners must read court cases that can be up to 100 pages lengthy just to sum up the most crucial details and determine whether their business should take on the case. As a result, Natural Language Processing (NLP) methods like information extraction and summarization provide excellent prospects to save law firms time.

Document summarization can reduce the amount of text that legal practitioners needed to understand the overall content of the document. This can help increase the efficiency for them in identifying relevant documents for their work through searching case abstracts instead of delving into lengthy documents.

In this example, we will perform text summarization of a litigation case published by the Securities and Exchange Commission about the manipulative trading of securities. It will be done using a model pre-trained on summarizing legal documents [9].

The text used was extracted from this website: https://www.sec.gov/news/press-release/2018-28

To generate the summary, we will have to import the model with its pre-trained weights, tokenize the text, and run it on our text.

```
from transformers import AutoTokenizer,␣
↪AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("nsi319/
↪legal-led-base-16384")
model = AutoModelForSeq2SeqLM.from_pretrained("nsi319/
↪legal-led-base-16384")
```

(continues on next page)

(continued from previous page)

```
padding = "max_length"



text="""On March 2, 2018, the Securities and Exchange␣
↪Commission announced securities fraud charges␣
↪against a U.K.-based broker-dealer and its␣
↪investment manager in connection with manipulative␣
↪trading in the securities of HD View 360 Inc., a U.
↪S.-based microcap issuer.  The SEC also announced␣
↪charges against HD View's CEO, another individual,␣
↪and three entities they control for manipulating HD␣
↪View's securities as well as the securities of␣
↪another microcap issuer, West Coast Ventures Group␣
↪Corp.  The SEC further announced the institution of␣
↪an order suspending trading in the securities of HD␣
↪View. These charges arise in part from an␣
↪undercover operation by the Federal Bureau of␣
↪Investigation, which also resulted in related␣
↪criminal prosecutions against these defendants by␣
↪the Office of the United States Attorney for the␣
↪Eastern District of New York. In a complaint filed␣
↪in the U.S. District Court for the Eastern District␣
↪of New York, the SEC alleges that Beaufort␣
↪Securities Ltd. and Peter Kyriacou, an investment␣
↪manager at Beaufort, manipulated the market for HD␣
↪View's common stock.  The scheme involved an␣
↪undercover FBI agent who described his business as␣
↪manipulating U.S. stocks through pump-and-dump␣
↪schemes.  Kyriacou and the agent discussed␣
↪depositing large blocks of microcap stock in␣
↪Beaufort accounts, driving up the price of the␣
↪stock through promotions, manipulating the stock's␣
↪price and volume through matched trades, and then␣
↪selling the shares for a large profit. The SEC's␣
↪complaint against Beaufort and Kyriacou alleges␣
↪that they: opened brokerage accounts for the␣
↪undercover agent in the names of nominees in order␣
↪to conceal his identity and his connection to the␣
↪anticipated trading activity in the accounts␣
↪suggested that the undercover agent could create␣
↪the false appearance that HD View's stock was ␣
```

(continues on next page)

(continued from previous page)

```
↪liquid in advance of a pump-and-dump by "gam[ing]␣
↪the market" through matched trades executed␣
↪multiple purchase orders of HD View shares with the␣
↪understanding that Beaufort's client had arranged␣
↪for an associate to simultaneously offer an␣
↪equivalent number of shares at the same priceA␣
↪second complaint filed by the SEC in the U.S.␣
↪District Court for the Eastern District of New York␣
↪alleges that in a series of recorded telephone␣
↪conversations with the undercover agent, HD View␣
↪CEO Dennis Mancino and William T. Hirschy agreed to␣
↪manipulate HD View's common stock by using the agent
↪'s network of brokers to generate fraudulent retail␣
↪demand for the stock in exchange for a kickback␣
↪from the trading proceeds.  According to the␣
↪complaint, the three men agreed that Mancino and␣
↪Hirschy would manipulate HD View stock to a higher␣
↪price before using the agent's brokers to liquidate␣
↪their positions at an artificially inflated price. ␣
↪The SEC's complaint also alleges that Mancino and␣
↪Hirschy executed a "test trade" on Jan. 31, 2018,␣
↪coordinated by the agent, consisting of a sell␣
↪order placed by the defendants filled by an␣
↪opposing purchase order placed by a broker into an␣
↪account at Beaufort.  Unbeknownst to Mancino and␣
↪Hirschy, the Beaufort account used for this trade␣
↪was a nominal account that was opened and funded by␣
↪the agent.  The SEC's complaint also alleges that,␣
↪prior to their contact with the undercover agent,␣
↪Mancino and Hirschy manipulated the market for HD␣
↪View and for West Coast by using brokerage accounts␣
↪that they owned, controlled, or were associated␣
↪with -including TJM Investments Inc., DJK␣
↪Investments 10 Inc., WT Consulting Group LLC - to␣
↪effect manipulative "matched trades."

... please refer to link provided for full text
"""
```

(continued from previous page)

```
input_tokenized = tokenizer.encode(text, return_
↪tensors='pt', padding=padding, pad_to_max_
↪length=True, max_length=6144, truncation=True)
summary_ids = model.generate(input_tokenized,
num_beams=4,
no_repeat_ngram_size=3,
length_penalty=2,
min_length=250,
max_length=500)
summary = [tokenizer.decode(g, skip_special_
↪tokens=True, clean_up_tokenization_spaces=False)␣
↪for g in summary_ids][0]
```

Now let us view what the model has generated.

```
print(summary)
```

```
On March 2, 2018, the Securities and Exchange␣
↪Commission announced securities fraud charges␣
↪against a U.K.-based broker-dealer and its␣
↪investment manager in connection with manipulative␣
↪trading in the securities of HD View 360 Inc.  The␣
↪SEC's complaint, filed in federal court in␣
↪Manhattan, alleges that Beaufort Securities Ltd.␣
↪and Peter Kyriacou, an investment manager at␣
↪Beaufort, manipulated the market for HD View's␣
↪common stock through pump-and-dump schemes. ␣
↪According to the complaint, the defendants opened␣
↪brokerage accounts for the undercover agent in the␣
↪names of nominees in order to conceal his identity␣
↪and his connection to the anticipated trading␣
↪activity in the stock, and then used the agent's␣
↪network of brokers to effect manipulative "matched␣
↪trades."  The complaint also alleges that, in a␣
↪series of recorded telephone conversations, Mancino␣
↪and Hirschy discussed depositing large blocks of␣
↪microcap stock in Beaufort accounts, driving up the␣
↪price of the stock through matched trades,␣
↪manipulating the stock's price and volume through␣
↪matched trading, and selling the shares for a large␣
↪profit.  In a parallel action, the U.S. Attorney's␣
↪Office for the Eastern District of New York␣
↪announced criminal charges against the defendants. ␣
↪On March 4, the SEC announced the filing of a civil␣
```

```
↪injunctive action against Beaufort.  The case is␣
↪being handled by Lara S. Mehraban, Robert Nesbitt,␣
↪and Antonia Chion of the SEC Enforcement Division's␣
↪Market Abuse Unit, with assistance from Joseph G.␣
↪Sansone of the Division of Economic and Risk␣
↪Analysis and the Office of the United States␣
↪Attorney's Retail Strategy Task Force. The SEC's␣
↪investigation was conducted by Preethi␣
↪Krishnamurthy and Ms. Shah and supervised by Cheryl␣
↪Crumpton. The SEC appreciates the assistance of the␣
↪Federal Bureau of Investigation, the Internal␣
↪Revenue Service, the Alberta Securities Commission,␣
↪and the Ontario Securities Commission in this␣
↪matter.
```

Let us see how much the model was able to summarize.

```
print("Summarized", len(text.split(" ")), "words into
↪", len(summary.split(" ")), "words")
```

```
Summarized 818 words into 301 words
```

The model was able to compress the article by 63% into 301 words.

### *15.3.2 Contract Review Assistant*

The traditional method of managing contracts requires a lot of manual effort and takes a lot of time [13]. Not to mention, it is a significant waste of the time that fee earners could use to charge clients and generate business for their firm.

Artificial intelligence can help alleviate this by helping to automate certain tasks. For instance, during a session of generating proposals, AI algorithms can learn to distinguish between a collection of non-disclosure agreements (NDAs). AI can identify NDAs in collections of documents and then design subsequent procedures in accordance with your needs. It can also help facilitate the process of contract checking by creating processes according to pre-defined data rules, such as "Send this contract to x employee for compliance checks if it is an NDA."

AI is able to uncover patterns in massive texts and find the connections between various contracts. Furthermore, it can spot mistakes and irregularities and develop patterns to improve business contracts. You can use it to extract useful information like the time to sign and frequently renegotiated contract terms.

This means that the information collected during the contract review process can be used to enhance procedures, assist renegotiation tactics, spot upselling possibilities, and gradually enhance operations.

In this example, we will be demonstrating how artificial intelligence can be used to assist in the contract review process by answering questions about the text within the contract provided [7, 8].

First let us begin by downloading the dataset in the link below [6]: https://github.com/TheAtticusProject/cuad/raw/main/data.zip. and unzip the contents into a folder named cuad-data.

Then now we will download the model weights from https://zenodo.org/record/4599830/files/roberta-base.zip?download=1 and unzip the contents into a folder named cuad-model.

We can then begin by importing the pre-trained model, tokenizer, and dataset for question and answers. The model has been previously trained on this dataset which has been curated for commercial contract understanding and will be able to answer questions about the contract.

```
import torch
from transformers import AutoConfig,␣
↪AutoModelForQuestionAnswering, AutoTokenizer
import json

model_path = "Business_Law/cuad-models/roberta-base/"
config_class, model_class, tokenizer_class =␣
↪(AutoConfig, AutoModelForQuestionAnswering,␣
↪AutoTokenizer)
config = config_class.from_pretrained(model_path)
tokenizer = tokenizer_class.from_pretrained(model_
↪path, do_lower_case=True, use_fast=False)
model = model_class.from_pretrained(model_path,␣
↪config=config)

with open('Business_Law/cuad-data/CUADv1.json') as␣
↪json_file:
    data = json.load(json_file)


questions = [i["question"] for i in data['data'][0][
↪'paragraphs'][0]['qas']][0:10]
paragraph = ' '.join(data['data'][0]['paragraphs'][0][
↪'context'].split())
```

Here is an example of a question that would be asked about the contract.

```
questions[2]
```

```
'Highlight the parts (if any) of this contract␣
↪related to "Agreement Date" that should be reviewed␣
↪by a lawyer. Details: The date of the contract'
```

This will be the contract paragraph that the model will have to look through to provide us with an answer.

```
paragraph[0:1000]
```

```
'EXHIBIT 10.6 DISTRIBUTOR AGREEMENT THIS DISTRIBUTOR␣
↪AGREEMENT (the "Agreement") is made by and between␣
↪Electric City Corp., a Delaware corporation (
↪"Company") and Electric City of Illinois LLC (
↪"Distributor") this 7th day of September, 1999.␣
↪RECITALS A. The Company\'s Business. The Company is␣
↪presently engaged in the business of selling an␣
↪energy efficiency device, which is referred to as␣
↪an "Energy Saver" which may be improved or␣
↪otherwise changed from its present composition (the
↪"Products"). The Company may engage in the business␣
↪of selling other products or other devices other␣
↪than the Products, which will be considered␣
↪Products if Distributor exercises its options␣
↪pursuant to Section~7 hereof. B. Representations.␣
↪As an inducement to the Company to enter into this␣
↪Agreement, the Distributor has represented that it␣
↪has or will have the facilities, personnel, and␣
↪financial capability to promote the sale and use of␣
↪Products. As an inducement to Distributor to enter␣
↪into this Agreement '
```

We can feed the question into the model together with the text that we want it to search from, in order to provide us with an answer.

```
# concatenates & encodes question and paragraph
encoding = tokenizer.encode_plus(text=questions[2],␣
↪text_pair=paragraph[0:1000])
# extracts the embeddings for model prediction
inputs = encoding['input_ids']
# get the tokens
tokens = tokenizer.convert_ids_to_tokens(inputs)

# make model prediction
outputs = model(input_ids=torch.tensor([inputs]))
```

(continues on next page)

(continued from previous page)

```
# get the start and end logits
start_scores = outputs.start_logits
end_scores = outputs.end_logits

# retrieve start & end tokens with the highest␣
↪probability
start_index = torch.argmax(start_scores)
end_index = torch.argmax(end_scores)

# retrieve the answer predicted by the model
answer = tokenizer.convert_tokens_to_␣
↪string(tokens[start_index:end_index+1]).strip()
print(answer.strip())
```

```
7th day of September, 1999.
```

As you can see, the model was able to determine the agreement date from the text provided.

However, you may have noticed that we only fed in the first 1000 characters from the contract into the model. Transformer models are trained with a fixed length input and cannot be fed more tokens than what it was trained on. Thus, we will need to work around it by splitting our inputs into multiple chunks so that our network will be able to process them one at a time and merge the results afterward.

```
import torch
import time
from torch.utils.data import DataLoader,␣
↪RandomSampler, SequentialSampler

from transformers import (
    AutoConfig,
    AutoModelForQuestionAnswering,
    AutoTokenizer,
    squad_convert_examples_to_features
)

from transformers.data.processors.squad import␣
↪SquadResult, SquadV2Processor, SquadExample
from transformers.data.metrics.squad_metrics import␣
↪compute_predictions_logits
```

We will set the following hyperparameters for our algorithm. These can be tuned according to different use cases.

```
### Setting hyperparameters
max_seq_length = 512
doc_stride = 256
n_best_size = 1
max_query_length = 64
max_answer_length = 512
do_lower_case = False
null_score_diff_threshold = 0.0
```

We will use the Squad Examples and Squad Results to help us process our dataset for large chunks of text. This is done by having the network predict whether each word is a start token or end token for the answer. To get this, the question will be repeated to the network at every segment of the text that was split from the contract.

```
examples = []

for i, question in enumerate(questions):
    example = SquadExample(
        qas_id=str(i),
        question_text=question,
        context_text=paragraph,
        answer_text=None,
        start_position_character=None,
        title="Predict",
        answers=None,
    )

    examples.append(example)

features, dataset = squad_convert_examples_to_
↪features(
    examples=examples,
    tokenizer=tokenizer,
    max_seq_length=max_seq_length,
    doc_stride=doc_stride,
    max_query_length=max_query_length,
    is_training=False,
    return_dataset="pt",
    threads=1,
)
```

(continues on next page)

(continued from previous page)

```
eval_sampler = SequentialSampler(dataset)
eval_dataloader = DataLoader(dataset, sampler=eval_
↪sampler, batch_size=10)
```

```
convert squad examples to features: 100
↪%|██████████| 10/10 [00:02<00:00,  3.43it/
↪s]
add example index and unique id: 100
↪%|██████████| 10/10 [00:00<00:00, 9967.
↪45it/s]
```

Now it is time for inference. We have split the dataset into smaller chunks of text, and now we will need to infer through all of them and aggregate the results. The final answer will be the sentence that lies between the highest start token score and the highest end token score.

```
all_results = []

device = torch.device("cuda" if torch.cuda.is_
↪available() else "cpu")
model.to(device)

for batch in eval_dataloader:
    model.eval()
    batch = tuple(t.to(device) for t in batch)

    with torch.no_grad():
        inputs = {
            "input_ids": batch[0],
            "attention_mask": batch[1],
            "token_type_ids": batch[2],
        }

        example_indices = batch[3]

        outputs = model(**inputs)

        for i, example_index in enumerate(example_
↪indices):
            eval_feature = features[example_index.
↪item()]
            unique_id = int(eval_feature.unique_id)
```

(continues on next page)

(continued from previous page)

```
            output = [output[i].detach().cpu().
↪tolist() for output in outputs.to_tuple()]

            start_logits, end_logits = output
            result = SquadResult(unique_id, start_
↪logits, end_logits)
            all_results.append(result)
```

Let us consolidate the answers from the logits.

```
answers = compute_predictions_logits(
    all_examples=examples,
    all_features=features,
    all_results=all_results,
    n_best_size=n_best_size,
    max_answer_length=max_answer_length,
    do_lower_case=do_lower_case,
    output_prediction_file=None,
    output_nbest_file=None,
    output_null_log_odds_file=None,
    verbose_logging=False,
    version_2_with_negative=True,
    null_score_diff_threshold=null_score_diff_
↪threshold,
    tokenizer=tokenizer
)
```

We will now predict for the first 10 questions in the dataset for this contract and visualize the answers generated.

```
for q, a in zip(questions, answers.values()):
    print("Question: {0}\nAnswer: {1}\n".format(q, a))
```

```
Question: Highlight the parts (if any) of this␣
↪contract related to "Document Name" that should be␣
↪reviewed by a lawyer. Details: The name of the␣
↪contract
Answer: DISTRIBUTOR AGREEMENT

Question: Highlight the parts (if any) of this␣
↪contract related to "Parties" that should be␣
↪reviewed by a lawyer. Details: The two or more␣
↪parties who signed the contract
```

(continues on next page)

(continued from previous page)

```
Answer: Electric City of Illinois L.L.C.


Question: Highlight the parts (if any) of this␣
↪contract related to "Agreement Date" that should be␣
↪reviewed by a lawyer. Details: The date of the␣
↪contract
Answer: 7th day of September, 1999.


Question: Highlight the parts (if any) of this␣
↪contract related to "Effective Date" that should be␣
↪reviewed by a lawyer. Details: The date when the␣
↪contract is effective
Answer: The term of this Agreement shall be ten (10)␣
↪years (the "Term") which shall commence on the date␣
↪upon which the Company delivers to Distributor the␣
↪last Sample, as defined hereinafter.


Question: Highlight the parts (if any) of this␣
↪contract related to "Expiration Date" that should␣
↪be reviewed by a lawyer. Details: On what date will␣
↪the contract's initial term expire?
Answer: The term of this Agreement shall be ten (10)␣
↪years (the "Term") which shall commence on the date␣
↪upon which the Company delivers to Distributor the␣
↪last Sample, as defined hereinafter.


Question: Highlight the parts (if any) of this␣
↪contract related to "Renewal Term" that should be␣
↪reviewed by a lawyer. Details: What is the renewal␣
↪term after the initial term expires? This includes␣
↪automatic extensions and unilateral extensions with␣
↪prior notice.
Answer: If Distributor complies with all of the terms␣
↪of this Agreement, the Agreement shall be renewable␣
↪on an annual basis for one (1) year terms for up to␣
↪another ten (10) years on the same terms and␣
↪conditions as set forth herein.
```

(continues on next page)

(continued from previous page)

```
Question: Highlight the parts (if any) of this␣
↪contract related to "Notice Period To Terminate␣
↪Renewal" that should be reviewed by a lawyer.␣
↪Details: What is the notice period required to␣
↪terminate renewal?
Answer: If Distributor complies with all of the terms␣
↪of this Agreement, the Agreement shall be renewable␣
↪on an annual basis for one (1) year terms for up to␣
↪another ten (10) years on the same terms and␣
↪conditions as set forth herein.


Question: Highlight the parts (if any) of this␣
↪contract related to "Governing Law" that should be␣
↪reviewed by a lawyer. Details: Which state/country
↪'s law governs the interpretation of the contract?
Answer: This Agreement is to be construed according␣
↪to the laws of the State of Illinois.


Question: Highlight the parts (if any) of this␣
↪contract related to "Most Favored Nation" that␣
↪should be reviewed by a lawyer. Details: Is there a␣
↪clause that if a third party gets better terms on␣
↪the licensing or sale of technology/goods/services␣
↪described in the contract, the buyer of such␣
↪technology/goods/services under the contract shall␣
↪be entitled to those better terms?
Answer:


Question: Highlight the parts (if any) of this␣
↪contract related to "Non-Compete" that should be␣
↪reviewed by a lawyer. Details: Is there a␣
↪restriction on the ability of a party to compete␣
↪with the counterparty or operate in a certain␣
↪geography or business or technology sector?
Answer: Term of the Agreement and for a period of␣
↪eighteen (18) months thereafter, nor will␣
↪Distributor solicit any customer or potential␣
↪customer of Company to purchase a competitive␣
↪product during that period.
```

### 15.3.3 Legal Research Assistant

In countries following the Common Law system (e.g., UK, USA, Canada, Australia, India), there are two primary sources of law—Statutes (established laws) and Precedents (prior cases). Statutes deal with applying legal principles to a situation (facts/scenario/circumstances which lead to filing the case). Precedents or prior cases help a lawyer understand how the court has dealt with similar scenarios in the past and prepare the legal reasoning accordingly.

When a lawyer is presented with a situation (which will potentially lead to filing of a case), it will be very beneficial to him/her if there is an automatic system that identifies a set of related prior cases involving similar situations as well as statutes/acts that can be most suited to the purpose in the given situation. Such a system shall not only help a lawyer but also benefit a common man, in a way of getting a preliminary understanding, even before he/she approaches a lawyer. It shall assist him/her in identifying where his/her legal problem fits, what legal actions he/she can proceed with (through statutes), and what were the outcomes of similar cases (through precedents).

In this exercise, we will be developing an artificial intelligence legal research assistant that helps to sort cases by their relevance [1]. This will help to increase the efficiency of lawyers when performing legal research as relevant cases are more likely to be presented to them first. To start, we will first have to download the dataset and place it in a folder named AILA_2019_dataset.
The dataset can be obtained from this website [10]:
https://www.kaggle.com/datasets/ananyapam7/legalai

Once we are done, we can begin by importing the relevant packages and datasets.

```
import glob
import functools
import datetime as dt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import random
import re
import numpy as np
import pandas as pd
import csv
```

```
read_files = glob.glob('Business_Law/AILA_2019_
↪dataset/Object_casedocs/*')
```

(continues on next page)

(continued from previous page)

```
with open("object_casedocs.csv", "w") as outfile:
    w=csv.writer(outfile)
    for f in read_files:
        with open(f, "r") as infile:
            w.writerow([" ".join([line.strip() for␣
↪line in infile])])

lst_arr = os.listdir('Business_Law/AILA_2019_dataset/
↪Object_casedocs/')
df_filename = pd.DataFrame(lst_arr, columns = ['Name
↪'])
df_filename
```

```
           Name
0        C1.txt
1       C10.txt
2      C100.txt
3     C1000.txt
4     C1001.txt
...         ...
2909   C995.txt
2910   C996.txt
2911   C997.txt
2912   C998.txt
2913   C999.txt

[2914 rows x 1 columns]
```

We will need to import the evaluation data which contains labels on whether a case document is relevant to a particular query document. The relevance will be marked as 0 if the document is irrelevant and 1 if it is relevant. This dataset should have a total of 195 relevant query–case pairs.

```
evaluate = pd.read_csv('Business_Law/AILA_2019_
↪dataset/relevance_judgments_priorcases.txt',␣
↪delimiter = " ", header = None)
evaluate.columns = ["Query_Number", "Q0", "Document" ,
↪"Relevance"]
evaluate=evaluate.drop(columns=["Q0"])
evaluate
```

```
       Query_Number Document  Relevance
0            AILA_Q1     C168          0
1            AILA_Q1     C382          0
2            AILA_Q1     C428          0
3            AILA_Q1     C949          0
4            AILA_Q1    C2303          0
...              ...      ...        ...
145695     AILA_Q50    C1367          0
145696     AILA_Q50    C2079          0
145697     AILA_Q50    C2066          0
145698     AILA_Q50    C1951          0
145699     AILA_Q50    C1111          0

[145700 rows x 3 columns]
```

Now let us import the case documents.

```
df = pd.read_csv('Business_Law/object_casedocs.csv',
↪header=None)
df.columns = ["Text"]
df
```

```
                                                   Text
0     Masud Khan v State Of Uttar Pradesh Supreme Co...
1     Prabhakaran Nair, Etc. v State Of Tamil Nadu A...
2     Hiten P. Dalal v Bratindranath Banerjee Suprem...
3     Ashok Kumar and Others v State of Tamil Nadu S...
4     Ashok Dhingra v N.C.T. of Delhi Supreme Court ...
...                                                 ...
2909  Rajendra Singh v State of Uttaranchal Supreme ...
2910  Food Corporation Of India & Anr v Seil Ltd. & ...
2911  State of Kerala v Sasi Supreme Court of India ...
2912  Columbia Sportswear Company v Director Of Inco...
2913  Bharat Gurjar and others v State of Rajasthan ...

[2914 rows x 1 columns]
```

We will need to know which text came from which case file, so we will concatenate the filenames to the text.

```
df = pd.concat([df_filename, df], axis = 1)
df
```

```
          Name                                               Text
0       C1.txt  Masud Khan v State Of Uttar Pradesh Supreme Co...
1      C10.txt  Prabhakaran Nair, Etc. v State Of Tamil Nadu A...
2     C100.txt  Hiten P. Dalal v Bratindranath Banerjee Suprem...
```

(continues on next page)

(continued from previous page)

```
3     C1000.txt  Ashok Kumar and Others v State of Tamil Nadu S...
4     C1001.txt  Ashok Dhingra v N.C.T. of Delhi Supreme Court ...
...         ...                                                ...
2909   C995.txt  Rajendra Singh v State of Uttaranchal Supreme ...
2910   C996.txt  Food Corporation Of India & Anr v Seil Ltd. & ...
2911   C997.txt  State of Kerala v Sasi Supreme Court of India ...
2912   C998.txt  Columbia Sportswear Company v Director Of Inco...
2913   C999.txt  Bharat Gurjar and others v State of Rajasthan ...

[2914 rows x 2 columns]
```

```
len(df)
```

```
2914
```

There is a total of 2914 case records.

Preprocessing the data

We first need to convert the texts to lowercase, remove punctuation, numbers, and special characters, using RegEx and strip. Numbers are removed as they often do not add much meaning to these texts apart from the dates.

```
import re
#Convert lowercase remove punctuation and Character␣
↪and then strip
text = df.iloc[0]["Text"]
text = re.sub(r'[^\w\s]', '', str(text).lower().
↪strip())
text = re.sub(r'\d+', '', text)
txt = text.split()
print(txt[0:100])
```

```
['masud', 'khan', 'v', 'state', 'of', 'uttar',␣
↪'pradesh', 'supreme', 'court', 'of', 'india', '26',␣
↪'september', '1973', 'writ', 'petition', 'no',␣
↪'117', 'of', '1973', 'the', 'judgment', 'was',␣
↪'delivered', 'by', 'a', 'alagiriswami', 'j', '1',␣
↪'petitioner', 'masud', 'khan', 'prays', 'for',␣
↪'his', 'release', 'on', 'the', 'ground', 'that',␣
↪'he', 'an', 'indian', 'citizen', 'has', 'been',␣
↪'illegally', 'arrested', 'and', 'confined', 'to',␣
↪'jail', 'under', 'paragraph', '5', 'of', 'the',␣
↪'foreigners', 'internment', 'order', '1962', 'he',␣
↪'had', 'come', 'to', 'india', 'from', 'pakistan',␣
↪'on', 'the', 'basis', 'of', 'a', 'pakistani',␣
↪'passport', 'dated', '1371954and', 'indian', 'visa',
↪ 'dated', '941956', 'in', 'his', 'application',␣
```

```
↪'for', 'visa', 'he', 'had', 'stated', 'that', 'he',␣
↪'had', 'migrated', 'to', 'pakistan', 'in', '1948',␣
↪'and', 'was', 'in']
```

The next thing we should do is to remove stopwords. Stopwords are words that do not provide much meaning to a text. These are removed to reduce wasting computation on these words and also ensure that the algorithms do not take them into account.

```
#remove stopwords
import nltk
nltk.download('stopwords')
lst_stopwords = nltk.corpus.stopwords.words("english")
txt = [word for word in txt if word not in lst_
↪stopwords]
print(txt[0:100])
```

```
['masud', 'khan', 'v', 'state', 'uttar', 'pradesh',
↪'supreme', 'court', 'india', '26', 'september',
↪'1973', 'writ', 'petition', '117', '1973', 'judgment
↪', 'delivered', 'alagiriswami', 'j', '1',
↪'petitioner', 'masud', 'khan', 'prays', 'release',
↪'ground', 'indian', 'citizen', 'illegally',
↪'arrested', 'confined', 'jail', 'paragraph', '5',
↪'foreigners', 'internment', 'order', '1962', 'come',
↪ 'india', 'pakistan', 'basis', 'pakistani',
↪'passport', 'dated', '1371954and', 'indian', 'visa',
↪ 'dated', '941956', 'application', 'visa', 'stated',
↪ 'migrated', 'pakistan', '1948', 'government',
↪'service', 'pakistan', 'pwd', 'darogha', 'given',
↪'permanent', 'address', 'hyderabad', 'sind', '2',
↪'statements', 'correct', 'petitioner', 'would',
↪'clearly', 'pakistani', 'national', 'fact', 'brought
↪', 'counter', 'affidavit', 'filled', 'behalf',
↪'respondent', 'petitioner', 'filed', 'affidavit',
↪'stating', 'appointed', 'police', 'constable',
↪'hasanganj', 'police', 'station', 'district',
↪'fatehpur', 'february', '1947', 'continued', 'police
↪', 'constable', 'till']
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_
↪data...
[nltk_data]   Package stopwords is already up-to-date!
```

Lastly, we will need to perform lemmatization. Lemmatization allows us to retrieve the base word of a provided word. This allows us to reduce the variation of the tokens used in the text as words with the same meaning would be grouped together. For example, run, runs, and running would all be lemmatized to run.

```
#Lemmetization
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

from nltk.corpus import wordnet as wn

def is_noun(tag):
    return tag in ['NN', 'NNS', 'NNP', 'NNPS']


def is_verb(tag):
    return tag in ['VB', 'VBD', 'VBG', 'VBN', 'VBP',
 ↪'VBZ']


def is_adverb(tag):
    return tag in ['RB', 'RBR', 'RBS']


def is_adjective(tag):
    return tag in ['JJ', 'JJR', 'JJS']


def penn_to_wn(tag):
    if is_adjective(tag):
        return wn.ADJ
    elif is_noun(tag):
        return wn.NOUN
    elif is_adverb(tag):
        return wn.ADV
    elif is_verb(tag):
        return wn.VERB
    return wn.NOUN

lem = nltk.stem.wordnet.WordNetLemmatizer()
print([lem.lemmatize(word, penn_to_wn(pos)) for word,␣
 ↪pos in nltk.pos_tag(txt)][0:100])
```

```
['masud', 'khan', 'v', 'state', 'uttar', 'pradesh',
↪'supreme', 'court', 'india', '26', 'september',
↪'1973', 'writ', 'petition', '117', '1973', 'judgment
↪', 'deliver', 'alagiriswami', 'j', '1', 'petitioner
↪', 'masud', 'khan', 'prays', 'release', 'ground',
↪'indian', 'citizen', 'illegally', 'arrest', 'confine
↪', 'jail', 'paragraph', '5', 'foreigner',
↪'internment', 'order', '1962', 'come', 'india',
↪'pakistan', 'basis', 'pakistani', 'passport', 'date
↪', '1371954and', 'indian', 'visa', 'date', '941956',
↪ 'application', 'visa', 'state', 'migrate',
↪'pakistan', '1948', 'government', 'service',
↪'pakistan', 'pwd', 'darogha', 'give', 'permanent',
↪'address', 'hyderabad', 'sind', '2', 'statement',
↪'correct', 'petitioner', 'would', 'clearly',
↪'pakistani', 'national', 'fact', 'bring', 'counter',
↪ 'affidavit', 'fill', 'behalf', 'respondent',
↪'petitioner', 'file', 'affidavit', 'state', 'appoint
↪', 'police', 'constable', 'hasanganj', 'police',
↪'station', 'district', 'fatehpur', 'february', '1947
↪', 'continued', 'police', 'constable', 'till']
```

Now let us put it all together in one function for pre-processing our dataset.

```
#to apply all the technique to all the records on␣
↪dataset
def utils_preprocess_text(text, flg_lemm =True, lst_
↪stopwords=None, remove_numbers = True ):
    text = re.sub(r'[^\w\s]', '', str(text).lower().
↪strip()) #\w\s

    if remove_numbers:
        text = re.sub(r'\d+', '', text)

    #tokenization(convert from string to List)
    lst_text = text.split()

    #remove stopwords
    if lst_stopwords is not None:
        lst_text = [word for word in lst_text if word␣
↪not in
                    lst_stopwords]

    #Lemmentization
```

(continues on next page)

(continued from previous page)

```
    if flg_lemm == True:
        lem = nltk.stem.wordnet.WordNetLemmatizer()
        lst_text = [lem.lemmatize(word, penn_to_
↪wn(pos)) for word, pos in nltk.pos_tag(lst_text)]

    # back to string from list
    text = " ".join(lst_text)
    return text
```

```
stopwords = nltk.corpus.stopwords.words("english")
CHARS_TO_REMOVE = list('''.()"',-:;''')
stopwords.extend(CHARS_TO_REMOVE)
```

Let us run our function on all the text and save the outputs under clean_text.

```
df['clean_text'] = df['Text'].apply(lambda x: utils_
↪preprocess_text(x, flg_lemm=True, lst_
↪stopwords=stopwords))
df.head(5)
```

```
        Name                                               Text  \
0     C1.txt  Masud Khan v State Of Uttar Pradesh Supreme Co...
1    C10.txt  Prabhakaran Nair, Etc. v State Of Tamil Nadu A...
2   C100.txt  Hiten P. Dalal v Bratindranath Banerjee Suprem...
3  C1000.txt  Ashok Kumar and Others v State of Tamil Nadu S...
4  C1001.txt  Ashok Dhingra v N.C.T. of Delhi Supreme Court ...

                                          clean_text
0  masud khan v state uttar pradesh supreme court...
1  prabhakaran nair etc v state tamil nadu or sup...
2  hiten p dalal v bratindranath banerjee supreme...
3  ashok kumar others v state tamil nadu supreme ...
4  ashok dhingra v nct delhi supreme court india ...
```

```
train = df["clean_text"]
train.head(5)
```

```
0    masud khan v state uttar pradesh supreme court...
1    prabhakaran nair etc v state tamil nadu or sup...
2    hiten p dalal v bratindranath banerjee supreme...
3    ashok kumar others v state tamil nadu supreme ...
4    ashok dhingra v nct delhi supreme court india ...
Name: clean_text, dtype: object
```

Now we will read in our query documents and perform the same pre-processing steps.

```
test = pd.read_csv("Business_Law/AILA_2019_dataset/
↪Query_doc.txt",delimiter = "|",header=None)
test.columns = ["AILA","NAN", "Query"]
test=test.drop(columns=["AILA","NAN"])
```

```
test['Query_processed'] = test['Query'].apply(lambda␣
↪x: utils_preprocess_text(x, flg_lemm=True, lst_
↪stopwords=stopwords))
```

```
test.head(5)
```

```
                                                Query  \
0  The appellant on February 9, 1961 was appointe...
1  The appellant before us was examined as prime ...
2  This appeal arises from the judgment of the le...
3  The Petitioner was married to the Respondent N...
4  This appeal is preferred against the judgment ...

                                     Query_processed
0  appellant february appoint officer grade iii r...
1  appellant u examine prime witness trial tr fil...
2  appeal arise judgment learn single judge high ...
3  petitioner marry respondent th november per hi...
4  appeal preferred judgment date pass high court...
```

After pre-processing out text, we will need to tokenize them into various tokens to be fed into our model. Tokenization will split the sentence into a sequence of words.

```
from nltk.tokenize import word_tokenize
nltk.download("punkt")
query_array_processed = [0]*len(test)

corpus_array_processed = [0]*len(train)

train_array=df.iloc[:, 2:].values

for i in range(len(train_array)):
    corpus_array_processed[i] = train_array[i][0]

query_array=test.iloc[:,1:].values

for i in range(len(query_array)):
    query_array_processed[i] = query_array[i][0]
```

(continues on next page)

(continued from previous page)

```
tokenized_corpus_array = [word_tokenize(_d.lower())␣
↪for _d in corpus_array_processed]
tokenized_query_array = [word_tokenize(_d.lower())␣
↪for _d in query_array_processed]
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

We will strip the ".txt" extension and the "C" in front of each case document so that we can easily compare the results.

```
ID = df["Name"]
ID = ID.str.rstrip('.txt').str.lstrip("C")
ID
```

```
0           1
1          10
2         100
3        1000
4        1001
         ...
2909      995
2910      996
2911      997
2912      998
2913      999
Name: Name, Length: 2914, dtype: object
```

For the retrieval algorithm, we will use BM25+ which is an improved variant of the Best Match 25 (BM25) ranking function. BM25 builds upon TF-IDF in order to better represent the relevance function. Like TF-IDF, BM25 rewards term frequency and penalizes document frequency but takes into account the limitations of its assumptions. For example, having twice the frequency of a high occurring word does not make a text twice as relevant. Furthermore, the length of a document compared to the mean document length should be put into consideration as well as more text but the same number of word occurrences would make a document less relevant.

We have set a k1 value higher at 10 as the text is significantly longer in legal applications and higher occurrences of a word is more significant.

```
from rank_bm25 import BM25Plus
bm25Plus = BM25Plus(tokenized_corpus_array, k1=10)
bm25Plus
```

```
<rank_bm25.BM25Plus at 0x208cf32ec40>
```

```
count = 0
topn = 15
for i in range(len(query_array_processed)):
    for j in bm25Plus.get_top_n(tokenized_query_
↪array[i], ID, n=topn):
        temp = evaluate.loc[evaluate['Query_Number']␣
↪== "AILA_Q"+str(i+1)]
        temp = temp[temp["Relevance"] == 1]["Document
↪"]
        for k in temp.str.replace('C', ''):
            if (j==k):
                count=count+1
count
```

```
38
```

```
Precision = count/(len(query_array_processed)*topn)
Recall = count/len(evaluate[evaluate["Relevance"]==1])

print(Precision)
print(Recall)
```

```
0.050666666666666665
0.19487179487179487
```

As you can see, by applying our algorithm, we can retrieve 19.5% of relevant texts by searching the top 15 documents out of 2914.

## Exercises

(1) Explain the difference between Limited Partnership and Limited Liability Partnership.
(2) Explain the difference between a C-corporation and an S-corporation.
(3) List at least five different types of business laws.
(4) Identify three ways that artificial intelligence could potentially be used in business law.

(5) Develop artificial intelligence solutions for predicting the judgement of a court. We will utilize this dataset [2]:
https://www.kaggle.com/datasets/deepcontractor/supreme-court-judgment-prediction
This dataset consists of information collected about various court proceedings and whether the first party won the case.
In this exercise, try to use what you have learnt so far to develop a model that can classify whether who won the case.

## References


1. Ananyapam7 (2021) GitHub-Ananyapam7/AILA-Artificial-Intelligence-for-Legal-Assistance: Python implementations of the various methods used in FIRE 2019 conference. https://github.com/Ananyapam7/AILA-Artificial-Intelligence-for-Legal-Assistance. Accessed 27 Apr 2023
2. Contractor D (2022) Supreme court judgment prediction. https://www.kaggle.com/datasets/deepcontractor/supreme-court-judgment-prediction
3. ContractsCounsel (2020) Business formation. https://www.contractscounsel.com/v/us/business-formation. Accessed 27 Apr 2023
4. Dale R (2021) Law and Word Order: NLP in Legal Tech. https://towardsdatascience.com/law-and-word-order-nlp-in-legal-tech-bd14257ebd06. Accessed 27 Apr 2023
5. Elawtalk (2022) What are the types of business law? https://elawtalk.com/types-of-business-law/. Accessed 27 Apr 2023
6. Hendrycks D, Burns C, Chen A, Ball S (2021) CUAD: an expert-annotated NLP dataset for legal contract review. https://github.com/TheAtticusProject/cuad/. In: CoRR. https://arxiv.org/abs/2103.06268, 2103.06268. Accessed 27 Apr 2023
7. Hotz H (2021a) How to set up a machine learning model for legal contract review. https://towardsdatascience.com/how-to-set-up-a-machine-learning-model-for-legal-contract-review-fe3b48b05a0e. Accessed 27 Apr 2023
8. Hotz H (2021b) How to set up a machine learning model for legal contract review—Part 2. https://towardsdatascience.com/how-to-set-up-a-machine-learning-model-for-legal-contract-review-part-2-6ecbbe680ba. Accessed 27 Apr 2023
9. Naren S (2021) nsi319/legal-led-base-16384 · Hugging Face. https://huggingface.co/nsi319/legal-led-base-16384. Accessed 27 Apr 2023
10. Paheli B, Kripabandhu G, Saptarshi G, Arindam P, Parth M, Arnab B, Prasenjit M (2019) Artificial intelligence for legal assistance (AILA). https://www.kaggle.com/datasets/ananyapam7/legalai. Accessed 27 Apr 2023
11. Toews R (2019) Ai will transform the field of law. https://www.forbes.com/sites/robtoews/2019/12/19/ai-will-transform-the-field-of-law/. Accessed 27 Apr 2023
12. UpCounsel (2023) Business law definition | UpCounsel 2023. https://www.upcounsel.com/business-law-definition. Accessed 27 Apr 2023
13. Zaremba Y (2022) Understanding the impact and role of AI in contract management. https://itchronicles.com/artificial-intelligence/understanding-the-impact-and-role-of-ai-in-contract-management/. Accessed 27 Apr 2023

# Chapter 16
# AI in Business Strategy


**Learning outcomes:**

- Be able to define what are business strategies.
- Be able to list examples of business strategies.
- Be able to list and understand various commonly used business strategy frameworks.
- Be able to list different examples of barriers to entry.
- Identify ways that artificial intelligence can be applied to business strategy.
- Develop artificial intelligence solutions for recommending acquisitions and identifying closest competitors and SWOT analysis.

## 16.1 Introduction to Business Strategy

A business strategy is an organized approach to accomplishing a company's overarching aims and objectives. It defines how a firm intends to compete in its selected market, including the products and services it plans to offer, intended customer demographics, and sales and marketing strategies. A business strategy may also include details about the organization's structure, alliances, and other factors that are important to the company's success. A business strategy's main objective is to develop a sustained competitive advantage for the organization.

### *16.1.1 Types of Business Strategies*

1. **Cost leadership**
   This strategy involves being the lowest cost producer in a market. Companies using this strategy aim to produce goods or services at a lower cost than their



T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0_16

competitors, which allows them to offer lower prices to customers. Walmart is a well-known example of a company that uses a cost leadership strategy.

2. **Differentiation**
   This strategy involves creating a unique product or service that is not offered by competitors. Companies using this strategy aim to differentiate themselves from competitors by offering something that is unique and valuable to customers. Apple is an example of a company that has used a differentiation strategy to great success.
3. **Market niche**
   This strategy involves targeting a specific, narrow market segment with specialized products or services. Companies using this strategy aim to become the best in their chosen niche, rather than trying to compete in a broader market. Tesla is an example of a company that has used a market niche strategy, focusing on electric cars and sustainable energy.
4. **Diversification**
   This strategy involves adding new products, services, or markets to a company's portfolio. Companies using this strategy aim to reduce risk by spreading their investments across different areas. GE is an example of a company that has used a diversification strategy, with businesses in industries such as aviation, healthcare, and energy.
5. **Structuralist**
   A structuralist business plan is founded on the existing market and industry standards. Everything, from goods to processes, is designed around the current market environments and industry standards, and the company strategy is built around it.
6. **Growth**
   Growth strategies are focused on increasing a company's size and market share. This can be accomplished through entering new markets, inventing new goods or services, or acquiring other businesses. Growth strategies can be risky but can also lead to increased market share and profitability.
7. **Price-skimming**
   Price-skimming strategies involve charging a high price for a product or service in the early stages of its life cycle. This can be an effective strategy for companies with a strong brand or a unique product that is in high demand. The goal of price-skimming is to maximize profits while demand is high before competition enters the market and prices drop.
8. **Acquisition**
   Acquisition strategies involve acquiring other companies to achieve strategic objectives, such as growth, cost savings, diversification, competitive advantage, or access to new technology or resources. Acquiring a company can be a fast and efficient way to achieve these objectives, but it can also be risky and requires careful planning and execution.

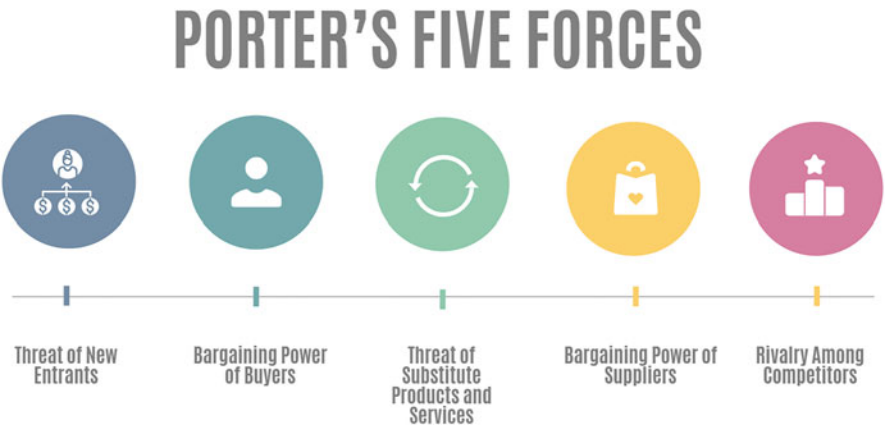



**Fig. 16.1** Porter's five forces

### *16.1.2 Business Strategy Frameworks*

Strategy frameworks are tools that help businesses structure their ideas and guide them as they grow and achieve their goals. They are often used to examine and build strategies. The following are some strategy frameworks which are commonly employed:

1. Porter's Five Forces: This framework as shown in Fig. 16.1, developed by Michael Porter, examines the competitive forces in an industry and how they impact a company's ability to generate profits. It can help a company to understand the competitive landscape and identify potential opportunities for differentiation or cost leadership.
2. BCG Matrix: The Boston Consulting Group (BCG) matrix, shown in Fig. 16.2, is a tool for analyzing a company's portfolio of products or business units. It lists various products on a chart of annual growth against market share. It helps a company to visualize which business units generate the most profits and which use the most resources. This information can be used to make strategic decisions about which business units to invest in, divest, or grow.
3. Value Chain Analysis: Value chain analysis, shown in Fig. 16.3, is a tool that helps to identify the activities that a company performs and the value that is added at each step. It helps a company to understand how it creates value and where it can create more value.
4. PESTLE analysis: PESTLE analysis shown in Fig. 16.4 is a tool that helps to identify the external factors that may impact a company. It looks at the political, economic, social, technological, legal, and environmental factors that can influence a business.
5. Product Life Cycle Analysis: Analyzing the product life cycle can help a business shape its strategy by providing insights into where a product or service is in its life cycle and what actions the business should take to maximize its revenue

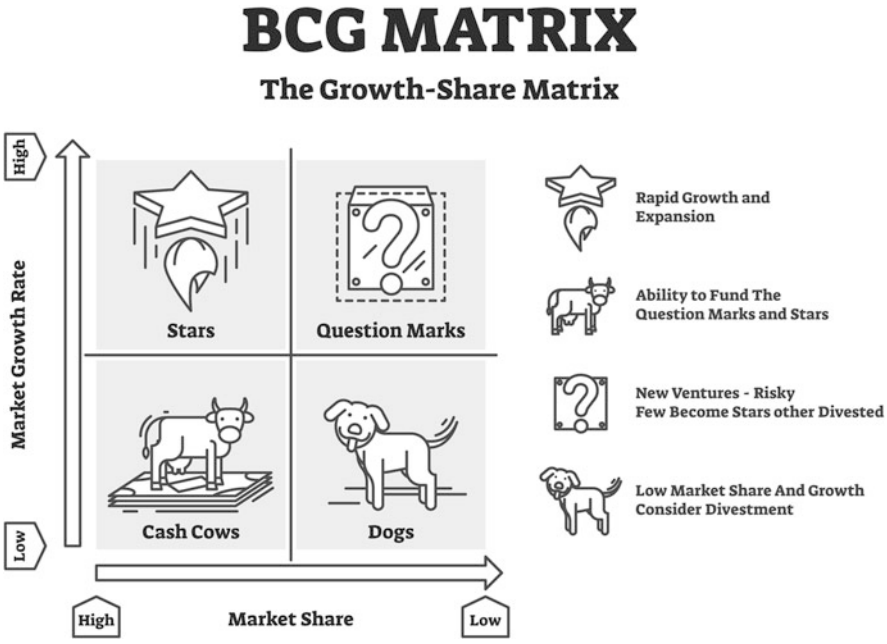



**Fig. 16.2** Boston consulting group matrix [6]

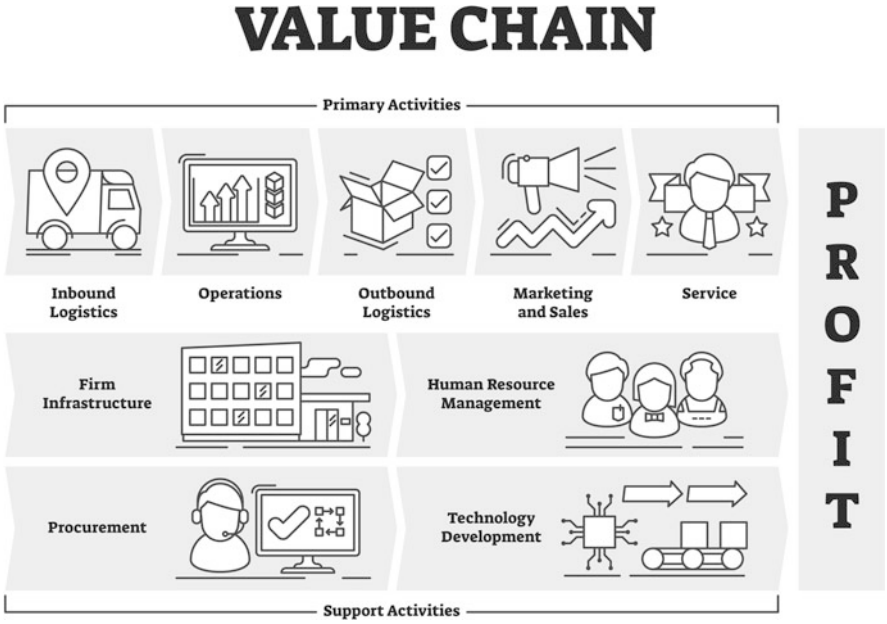



**Fig. 16.3** Value chain analysis [10]

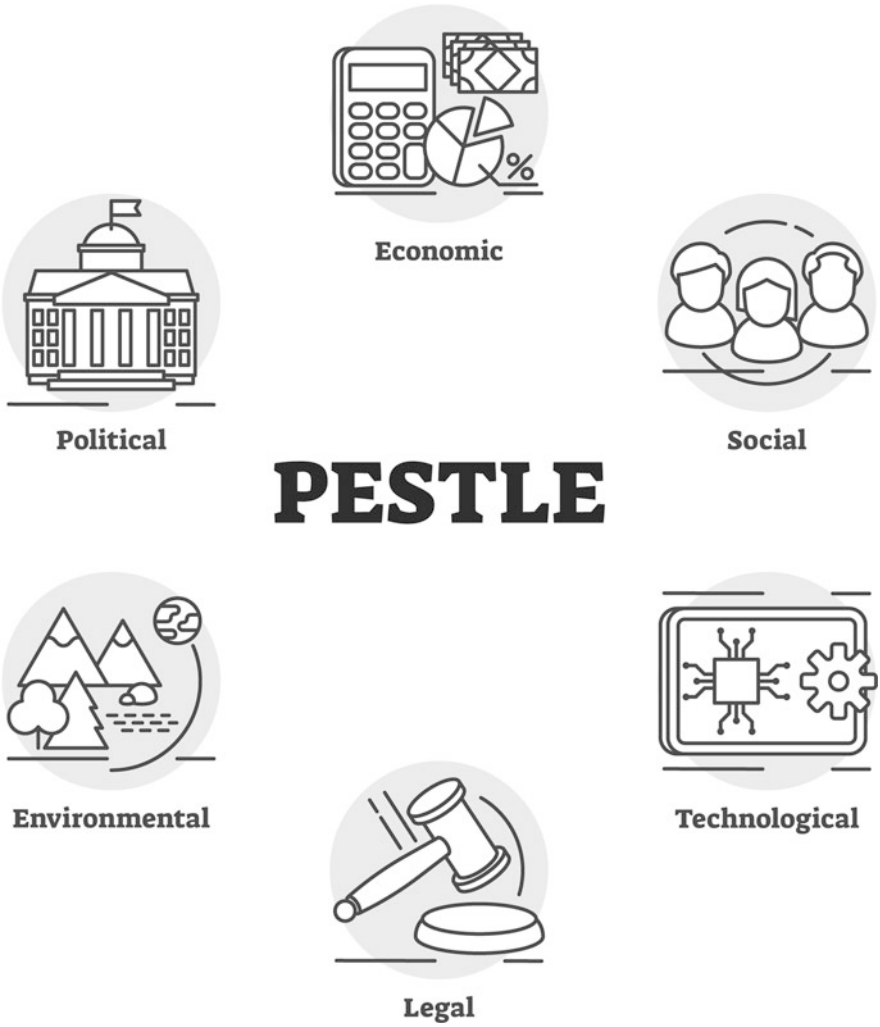



**Fig. 16.4** PESTLE analysis [8]

and profits. By understanding where a product or service is in its life cycle, a business can make more informed decisions about how to allocate resources and make strategic changes to optimize revenue and profits. Figure 16.5 shows the different stages of a product's life cycle.

6. SWOT analysis: SWOT analysis, shown in Fig. 16.6, is a tool that can be used to evaluate a company's internal strengths and weaknesses, as well as external opportunities and threats. It is often used as a starting point for developing a business strategy, by providing an overview of a company's current position in the market.

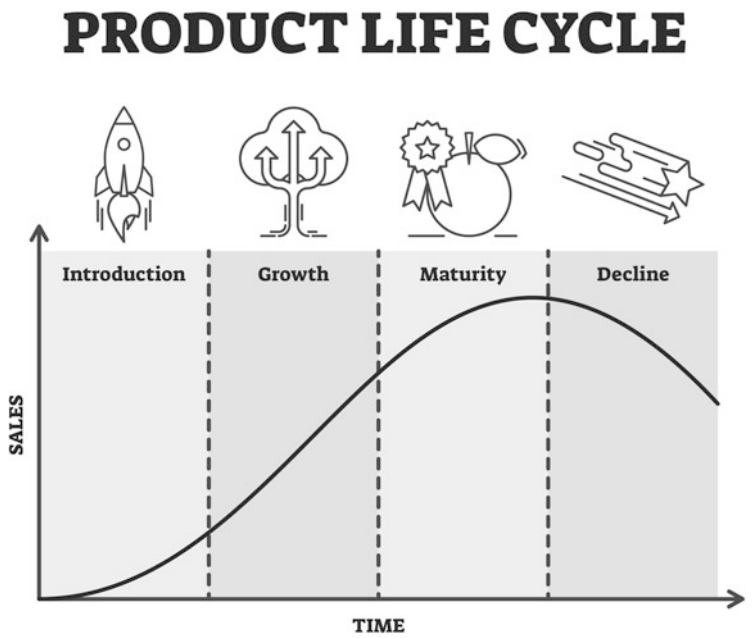



**Fig. 16.5** Product life cycle analysis [9]

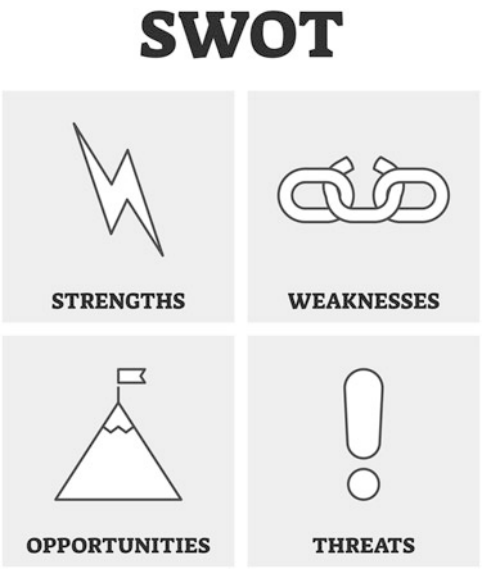



**Fig. 16.6** SWOT analysis [7]

All these frameworks are designed to help companies to understand their internal and external environment, identify opportunities and risks, and make informed decisions about their business strategy. The choice of which framework to use depends on the company's goals and what it wants to achieve.

### *16.1.3 Barriers to Entry*

Barriers to entry refer to the obstacles that make it difficult for new companies to enter a market and compete with established players. The higher the barriers to entry, the more difficult it is for new companies to enter the market and compete. High barriers to entry can help protect a company's competitive advantages by making it difficult for new companies to enter the market and compete. This allows established companies to maintain their dominant position and continue to reap the benefits of their competitive advantages.

Companies can develop various barriers to entry to deter potential threats from other companies. Some examples of these are:

1. **Technical barriers**
   Developing and maintaining a high-quality, reliable, and scalable solution requires significant technical expertise and resources. This can be a significant barrier to entry for new companies, as they may not have the necessary experience or resources to develop a competitive product.
2. **Economies of scale**
   Companies that have already established a large customer base can benefit from economies of scale, which can make it difficult for new companies to compete on price.
3. **Network effects**
   In some cases, the value of a product increases as more customers use it. This creates a network effect, which can make it difficult for new companies to attract customers, as their product may not be as valuable without a large customer base.
4. **Brand recognition**
   Established companies often have a strong brand recognition and reputation, which can make it difficult for new companies to attract customers.
5. **Capital requirements**
   Developing and scaling certain businesses can be capital-intensive, and new companies may not have the resources or access to funding to compete with established players.
6. **Intellectual Property**
   Strong patents, trademarks, and copyrights can protect a company's products or processes, making it difficult for new companies to enter the market.

7. **Customer Relationships**
   A company that has established strong relationships with its customers, such as through loyalty programs, can have a significant advantage over new entrants, as it can be difficult for new companies to attract these customers.
8. **Government Regulations**
   Certain industries, such as healthcare and energy, are heavily regulated, and the process of obtaining licenses, permits, and approvals can be a significant barrier to entry for new companies.

## 16.2 Artificial Intelligence in Business Strategy

Artificial intelligence has demonstrated magnificent results in strategy development. Games like chess and go have been tackled by artificial intelligence with amazing results, demonstrating strong potential in its application in strategy development. In fact, artificial intelligence has been applied to model and monitor the strategic decision-making of governments and corporations. One such example is a machine learning model based on China's decision-making called the Policy Change Index which predicts the most relevant issues faced by the country, based on the text provided by the People's Daily newspapers [4]. This information highlights what current issues are the government most concerned with and hence would provide clues to how the Chinese government will be allocating its resources.

However, completely automating corporate strategies using artificial intelligence is still far away in the future. Artificial intelligence researchers are working on science fiction concepts such as artificial general intelligence, but it is a long way to go before these ideas get realized and that form of AI is simply irrelevant to the present. While many use cases of artificial intelligence show promise in strategy, it generally works well in conditions where the consequences of each action are known before making a decision. However, it is only until recently that artificial intelligence has had some success in making decisions under incomplete information scenarios. Furthermore, understanding how complex artificial intelligence algorithms make decisions is still largely unsolved. Furthermore, strategic decisions have enormous repercussions. Thus, it is imperative for executives to be able to understand the reasons behind its predictions and identify which set of data is it extrapolating from. Only then they are able to assess whether they can trust these predictions.

While it is not possible to perform a high level of automation on strategic business decisions, there are many possibilities for incorporating AI into strategy building blocks to significantly improve the decision-making process. Artificial intelligence can be used to assist and crawl through large amounts of data points from various sources such as reviews, financial statements, and social media to analyze the competitive landscape businesses face. Through such use cases, artificial intelligence can help provide useful insights to business leaders for strategic decision-making. In this chapter, we will go through some examples of use cases that artificial intelligence can bring.

## 16.3 Applications of AI in Business Strategy

### *16.3.1 Startup Acquisition*

Mergers and acquisitions (M&A) activity has been demonstrated to be highly linked with corporate performance and is becoming an increasingly important aspect of a successful business strategy in today's society. Acquiring successful startups can bring many benefits to larger businesses. By capitalizing on the strengths of the acquired company's targets, the acquiring company can expand its primary business or access brand new markets for revenue growth. These are some of the reasons that acquiring a startup can be a good business strategy [2]:

1. **Filling a product gap**
   Rather than developing technology from scratch, businesses frequently acquire and integrate it into their product pipeline. In order to complement their primary offering, businesses frequently purchase an anchor component in a new product category. Acquisitions are one strategy that a corporation may employ to improve the primary goods or services that it provides to customers.
2. **Expanding the customer base**
   A corporation can expand through the acquisition of a target with a relevant consumer base. This can assist the organization reach new markets and increase the reputation of the product/service, allowing it to achieve a greater market share.
3. **Recruiting domain experts**
   Possessing the appropriate competencies may provide a significant edge over one's competitors and should be a priority for any business. By acquiring experienced personnel, M&A may improve the acquirer's internal capabilities and core product. These are relatively minor acquisitions that will complement the acquirer's main digital business strategy and skills.

For companies that are looking to use strategic acquisitions as a business strategy, monitoring and identifying successful startups to acquire would be highly important. However, investigating whether a company is worth acquiring requires a lot of research and can be very time-consuming. It becomes even more difficult as identifying suitable acquisition targets can be very challenging when there are hundreds of thousands of startups across the world. In this aspect, predicting startup success can be useful to pre-filter potential acquisition targets and facilitate strategic acquisitions.

In this example, we will work on predicting the likelihood that a startup is worth acquiring [1].

The data can be obtained from [3]:
https://www.kaggle.com/datasets/manishkc06/startup-success-prediction

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV,␣
↪train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
```

Let us read the data.

```
df=pd.read_csv("Business_Strategy/startup data.csv")

df.head()
```

```
   Unnamed: 0 state_code   latitude   longitude zip_code       id  \
0        1005         CA  42.358880  -71.056820    92101   c:6669
1         204         CA  37.238916 -121.973718    95032  c:16283
2        1001         CA  32.901049 -117.192656    92121  c:65620
3         738         CA  37.320309 -122.050040    95014  c:42668
4        1002         CA  37.779281 -122.419236    94105  c:65806

            city              Unnamed: 6               name  labels  ...  \
0      San Diego                     NaN        Bandsintown       1  ...
1      Los Gatos                     NaN          TriCipher       1  ...
2      San Diego      San Diego CA 92121              Plixi       1  ...
3      Cupertino      Cupertino CA 95014  Solidcore Systems       1  ...
4  San Francisco  San Francisco CA 94105     Inhale Digital       0  ...

  object_id has_VC has_angel has_roundA  has_roundB  has_roundC  has_roundD  \
0    c:6669      0         1          0           0           0           0
1   c:16283      1         0          0           1           1           1
2   c:65620      0         0          1           0           0           0
3   c:42668      0         0          0           1           1           1
4   c:65806      1         1          0           0           0           0

   avg_participants  is_top500    status
0            1.0000          0  acquired
1            4.7500          1  acquired
2            4.0000          1  acquired
3            3.3333          1  acquired
4            1.0000          1    closed

[5 rows x 49 columns]
```

Now let us see what columns are there.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 923 entries, 0 to 922
Data columns (total 49 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Unnamed: 0                 923 non-null    int64
 1   state_code                 923 non-null    object
 2   latitude                   923 non-null    float64
 3   longitude                  923 non-null    float64
 4   zip_code                   923 non-null    object
 5   id                         923 non-null    object
 6   city                       923 non-null    object
 7   Unnamed: 6                 430 non-null    object
 8   name                       923 non-null    object
 9   labels                     923 non-null    int64
 10  founded_at                 923 non-null    object
 11  closed_at                  335 non-null    object
 12  first_funding_at           923 non-null    object
 13  last_funding_at            923 non-null    object
 14  age_first_funding_year     923 non-null    float64
 15  age_last_funding_year      923 non-null    float64
 16  age_first_milestone_year   771 non-null    float64
 17  age_last_milestone_year    771 non-null    float64
 18  relationships              923 non-null    int64
 19  funding_rounds             923 non-null    int64
 20  funding_total_usd          923 non-null    int64
 21  milestones                 923 non-null    int64
 22  state_code.1               922 non-null    object
 23  is_CA                      923 non-null    int64
 24  is_NY                      923 non-null    int64
 25  is_MA                      923 non-null    int64
 26  is_TX                      923 non-null    int64
 27  is_otherstate              923 non-null    int64
 28  category_code              923 non-null    object
 29  is_software                923 non-null    int64
 30  is_web                     923 non-null    int64
 31  is_mobile                  923 non-null    int64
 32  is_enterprise              923 non-null    int64
 33  is_advertising             923 non-null    int64
 34  is_gamesvideo              923 non-null    int64
 35  is_ecommerce               923 non-null    int64
 36  is_biotech                 923 non-null    int64
 37  is_consulting              923 non-null    int64
 38  is_othercategory           923 non-null    int64
 39  object_id                  923 non-null    object
 40  has_VC                     923 non-null    int64
 41  has_angel                  923 non-null    int64
 42  has_roundA                 923 non-null    int64
 43  has_roundB                 923 non-null    int64
```

(continues on next page)

(continued from previous page)

```
 44  has_roundC                923 non-null    int64
 45  has_roundD                923 non-null    int64
 46  avg_participants          923 non-null    float64
 47  is_top500                 923 non-null    int64
 48  status                    923 non-null    object
dtypes: float64(7), int64(28), object(14)
memory usage: 353.5+ KB
```

Let us drop the irrelevant columns such as location information.

```
df.drop(["Unnamed: 0","Unnamed: 6","id","state_code",
 ↪"category_code", "state_code.1","latitude",
 ↪"longitude","zip_code","city","name",
       "closed_at","founded_at","first_funding_at",
 ↪"last_funding_at","object_id"], axis=1,␣
 ↪inplace=True)

df.head()
```

```
   labels  age_first_funding_year  age_last_funding_year  \
0       1                  2.2493                 3.0027
1       1                  5.1260                 9.9973
2       1                  1.0329                 1.0329
3       1                  3.1315                 5.3151
4       0                  0.0000                 1.6685

   age_first_milestone_year  age_last_milestone_year  relationships  \
0                    4.6685                   6.7041              3
1                    7.0055                   7.0055              9
2                    1.4575                   2.2055              5
3                    6.0027                   6.0027              5
4                    0.0384                   0.0384              2

   funding_rounds  funding_total_usd  milestones  is_CA  ...  \
0               3             375000           3      1  ...
1               4           40100000           1      1  ...
2               1            2600000           2      1  ...
3               3           40000000           1      1  ...
4               2            1300000           1      1  ...

   is_othercategory  has_VC  has_angel  has_roundA  has_roundB  has_roundC  \
0                 1       0          1           0           0           0
1                 0       1          0           0           1           1
2                 0       0          0           1           0           0
3                 0       0          0           0           1           1
4                 0       1          1           0           0           0

   has_roundD  avg_participants  is_top500    status
0           0            1.0000          0  acquired
1           1            4.7500          1  acquired
2           0            4.0000          1  acquired
3           1            3.3333          1  acquired
4           0            1.0000          1    closed

[5 rows x 33 columns]
```

As we can see from the data, label 1, indicates that a startup was successful and was acquired by another company, whereas 0 means that a startup was not successful.

Now let us check out data for empty values.

```
df.isnull().sum().sort_values(ascending=False).head()
```

```
age_first_milestone_year    152
age_last_milestone_year     152
labels                        0
has_angel                     0
is_ecommerce                  0
dtype: int64
```

We have 152 empty values in both the "age_first_milestone_year" and "age_last_milestone_year" columns. We will fill these empty values with the median values of the column.

```
df.age_first_milestone_year.fillna(df["age_first_
↪milestone_year"].median(), inplace=True)
df.age_last_milestone_year.fillna(df["age_last_
↪milestone_year"].median(), inplace=True)
```

Now we need to separate the features from the ground truth labels.

```
X=df.drop(["status", "labels"], axis=1)
Y=df.labels
```

We will split our dataset into training and testing with a ratio of 70% in training and 30% in testing.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,
↪ Y, test_size=0.30, random_state=42)
```

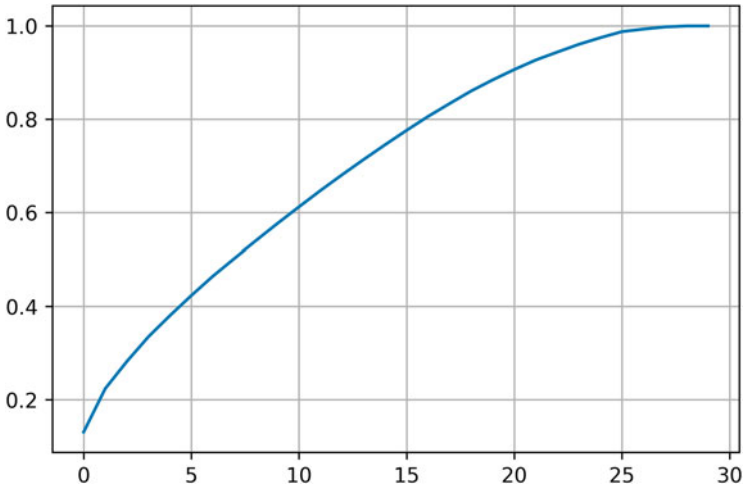
We will normalize each feature to have mean 0 and standard deviation 1. This will allow our features to equally weighted when using PCA to perform dimensionality reduction.

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

We can see how much of the original data is explained by the number of principal components used in the graph below. As you can see, the last few principal components do not contribute as much after the 90th percentile.

```
pca=PCA(n_components=len(X.columns)-1)

X_pca=pca.fit_transform(X_train)
exp_var=pca.explained_variance_ratio_
cumsum_var=np.cumsum(exp_var)
print(cumsum_var)
plt.plot(cumsum_var)
plt.grid()
```

```
[0.1308337  0.22365973 0.28174192 0.3346305  0.
↪38009515 0.42333056
 0.46477985 0.50263267 0.53999606 0.57659398 0.
↪6122668  0.6472743
 0.68087381 0.71354846 0.74537942 0.77623246 0.
↪80610559 0.83372807
 0.86063452 0.88447349 0.90656267 0.92701329 0.
↪94402292 0.96063281
 0.97485717 0.98784006 0.99323711 0.99786195 0.
↪99992513 1.        ]
```

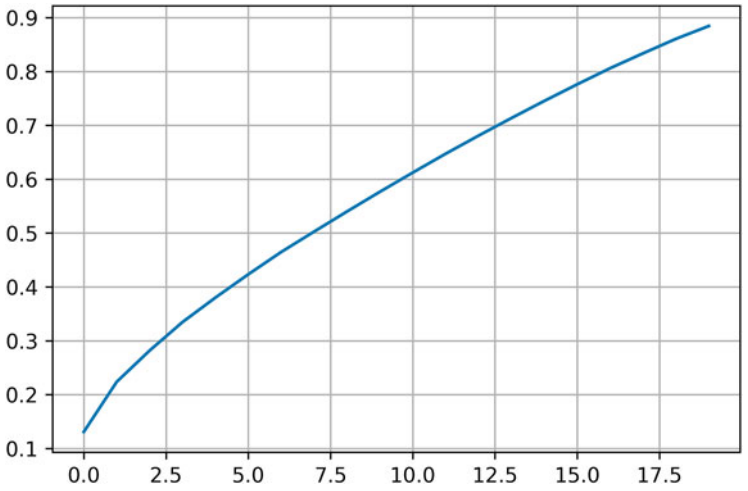



```
pca_new=PCA(n_components=(cumsum_var > 0.9).
↪nonzero()[0][0])
X_new=pca_new.fit_transform(X_train)

exp_var_new=pca_new.explained_variance_ratio_
cumsum_var_new=np.cumsum(exp_var_new)
```

(continues on next page)

(continued from previous page)

```
plt.plot(cumsum_var_new)
plt.grid()
X_new=pd.DataFrame(X_new)
```




Let us use PCA to transform our features.

```
X_train_pca = pca_new.transform(X_train)
X_test_pca = pca_new.transform(X_test)
```

Now we can train our model. We will train the random forest classifier algorithm to predict the success of the startups and evaluate it on the test set.

```
forest=RandomForestClassifier(n_estimators=20,
                              max_depth=6,
                              criterion="gini",
                               random_state=42,
                              )

forest.fit(X_train,Y_train)
y_test_pred_forest=forest.predict(X_test)
y_train_pred_forest=forest.predict(X_train)

print("Classification Report:","\n",classification_
 ↪report(Y_test,y_test_pred_forest),"\n")
```

(continues on next page)

(continued from previous page)

```
print("Confusion Matrix","\n",confusion_matrix(Y_test,
↪y_test_pred_forest),"\n")
print("Accuracy score of random forest test set:",
↪accuracy_score(Y_test,y_test_pred_forest))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.60      0.70       100
           1       0.80      0.93      0.86       177

    accuracy                           0.81       277
   macro avg       0.82      0.77      0.78       277
weighted avg       0.82      0.81      0.80       277


Confusion Matrix
 [[ 60  40]
 [ 12 165]]

Accuracy score of random forest test set: 0.8122743682310469
```

Let us predict the probability of success for the first 10 startups in the test set.

```
for prob, ground_truth in zip(forest.predict_proba(X_
↪test[0:10,:])[:,1], Y_test[0:10]):
    print(round(prob,2), ground_truth)
```

```
0.82 1
0.57 1
0.71 0
0.38 0
0.47 0
0.04 0
0.7 1
0.72 1
0.77 0
0.69 1
```

Now let us rank the startups by most likely to become successful and identify the top 5 most worth looking into, according to our model.

(continued from previous page)

```
probability_of_success = forest.predict_proba(X_
↪test)[:,1]
print("Top 5 startups most likely to succeed in the␣
↪test set:", probability_of_success.argsort()[::-
↪1][0:5])
print("With probabilities of:", probability_of_
↪success[probability_of_success.argsort()[::-
↪1][0:5]])
```

```
Top 5 startups most likely to succeed in the test␣
↪set: [180 102 169 117  57]
With probabilities of: [0.94758606 0.94215402 0.
↪937314   0.93586826 0.93088208]
```

### *16.3.2 Identifying Closest Competitors*

Brands are often compared against one another in various reviews, and this can provide insights to business leaders on which companies do they have to watch out for the most within their industry. In this example we will map out the competitive landscape and see which brands are often mentioned together in reviews [5].

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from bs4 import BeautifulSoup
import requests
import nltk
import nltk.corpus
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

The model details table lists which brand does a car model belong to.

```
model_details = pd.read_csv(r"Business_Strategy\Text-
↪Mining-Car-Review-Data\models.csv")
model_details.head()
```

| | brand | model |
|---|---|---|
| 0 | acura | integra |
| 1 | acura | Legend |
| 2 | acura | vigor |
| 3 | acura | rlx |
| 4 | acura | ILX |

The attributes table categorizes various key words into topic categories.

```
attributes = pd.read_csv(r".\Text-Mining-Car-Review-
↪Data\attributes.csv")
attributes.head()
```

| | category | word |
|---|---|---|
| 0 | cost | price |
| 1 | cost | value |
| 2 | cost | warranty |
| 3 | performance | engine |
| 4 | performance | power |

The aspirational table lists words that represent interest in purchasing a particular model.

```
# import aspirational data
aspirational = pd.read_csv(r"Business_Strategy\Text-
↪Mining-Car-Review-Data\aspirational.csv")
aspirational.head()
```

```
       category      word
0  aspirational       buy
1  aspirational      sale
2  aspirational  consider
3  aspirational    dealer
4  aspirational     offer
```

In this example, we have extracted text reviews from this forum: https://forums.edmunds.com/discussion/7526/general/x/midsize-sedans-2-0

The forum discusses on various mid-size sedan models and brands which provides useful insights on how people view the various car brands for the mid-size sedan category.

```
df = pd.read_csv(r"Business_Strategy\Text-Mining-Car-
↪Review-Data\edmunds_extraction.csv")
df.head()
```

```
   Counter                    Date              User  \
0        1   April 11, 2007 6:52PM     \r\nmotownusa
1        2   April 11, 2007 7:33PM      \r\nexshoman
2        3   April 12, 2007 6:51AM  \r\ntargettuning
3        4   April 12, 2007 8:43AM           \r\npat
4        5  April 13, 2007 11:49AM         \r\nperna

                                          Comment
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we're here...
4  \r\nThe Altima is my favorite of the bunch. It...
```

```
# Drop any rows with null values
df.dropna(axis=0, inplace=True)
df.shape
```

```
(5000, 4)
```

We will begin preprocessing the reviews by decontracting contractions by converting words such as “won’t” into “will not.”

```
#expand contraction words
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
```

(continues on next page)

(continued from previous page)

```
    phrase = re.sub(r"can\'t", "can not", phrase)
    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

df['Comment1'] = df.apply(lambda row : decontracted(row['Comment']), axis = 1)
```

Next we will remove trailing whitespaces and full stops and perform tokenization to split the reviews into smaller tokens for text analysis.

```
#Tokenization the comments column
def token_(x):
    x = x.strip(' \r.\n')
    token = word_tokenize(x)
    return token
df['Comment_token'] = df.apply(lambda row : token_(row['Comment1']), axis = 1)
```

```
df['Comment_token']
```

```
0       [Hi, Pat, :, You, forgot, the, Chrysler, Sebring]
1       [I, am, sure, some, folks, would, appreciate, ...
2       [You, can, try, to, revive, this, topic, but, ...
3       [Model, vs., model, is, exactly, what, we, are...
4       [The, Altima, is, my, favorite, of, the, bunch...
                              ...
4995    [``, Let, me, try, one, more, time, ., Accord,...
4996    [``, No, one, likes, repairs, on, their, car, ...
4997    [Toyota, &, GM, gives, you, guys, what, they, ...
4998    [What, weaknesses, ?, you, have, goota, be, ki...
4999    [Well, the, cheapest, service, did, not, have,...
Name: Comment_token, Length: 5000, dtype: object
```

The text comes in various cases which is not ideal as the same word could be captured differently due to capitalization. Hence, we will convert all letters to its lower case.

```
# Lower Casing the Tokenized comments
def lower_case(x):
    ret = []
    for words in x:
        words = words.lower()
        ret.append(words)
    return ret
df['Comment_token'] = df.apply(lambda row : lower_
↪case(row['Comment_token']), axis = 1)
```

```
df.head()
```

```
   Counter                    Date                User  \
0        1   April 11, 2007 6:52PM     \r\nmotownusa
1        2   April 11, 2007 7:33PM      \r\nexshoman
2        3   April 12, 2007 6:51AM  \r\ntargettuning
3        4   April 12, 2007 8:43AM           \r\npat
4        5  April 13, 2007 11:49AM         \r\nperna

                                          Comment  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we're here...
4  \r\nThe Altima is my favorite of the bunch. It...

                                         Comment1  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI am sure some folks would appreciate havi...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we are her...
4  \r\nThe Altima is my favorite of the bunch. It...

                                      Comment_token
0  [hi, pat, :, you, forgot, the, chrysler, sebring]
1  [i, am, sure, some, folks, would, appreciate, ...
2  [you, can, try, to, revive, this, topic, but, ...
3  [model, vs., model, is, exactly, what, we, are...
4  [the, altima, is, my, favorite, of, the, bunch...
```

We will do the same for the model details table.

```
def lower(x):
    l = x.lower()
    return l
# Applying the 'lower' function to model_details␣
↪dataframe in both the columns
```

(continues on next page)

(continued from previous page)

```
model_details['brand'] = model_details.apply(lambda␣
↪row : lower(row['brand']), axis = 1)
model_details['model'] = model_details.apply(lambda␣
↪row : lower(row['model']), axis = 1)
```

In order to determine how the overall brand is viewed between mid-size sedans, we will first begin by replacing all of the car models discussed by their corresponding car brands. This will allow us to aggregate the information provided across various car models in a particular brand.

```
#Replacing the model names with brand names
for i in range(len(df)):
    for j in range(len(df['Comment_token'][i])):
        if df['Comment_token'][i][j] in model_details[
↪'model'].tolist():
            df['Comment_token'][i][j] = model_details[
↪'brand'][model_details['model'].tolist().index(df[
↪'Comment_token'][i][j])]
```

```
df.head()
```

```
   Counter                    Date                User  \
0        1   April 11, 2007 6:52PM     \r\nmotownusa
1        2   April 11, 2007 7:33PM      \r\nexshoman
2        3   April 12, 2007 6:51AM  \r\ntargettuning
3        4   April 12, 2007 8:43AM           \r\npat
4        5  April 13, 2007 11:49AM         \r\nperna

                                             Comment  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we're here...
4  \r\nThe Altima is my favorite of the bunch. It...

                                            Comment1  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI am sure some folks would appreciate havi...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we are her...
4  \r\nThe Altima is my favorite of the bunch. It...

                                       Comment_token
0  [hi, pat, :, you, forgot, the, chrysler, chrys...
1  [i, am, sure, some, folks, would, appreciate, ...
2  [you, can, try, to, revive, this, topic, but, ...
3  [model, vs., model, is, exactly, what, we, are...
4  [the, nissan, is, my, favorite, of, the, bunch...
```

Now we will remove the punctuations from the comments as well and delete all tokens containing empty strings.

```
# Removing Punctuation
import re
punctuation = re.compile(r'[-.?!,:;()%\/|0-9""]')
def post_punctuation(x):
    ret = []
    for words in x:
        item = punctuation.sub("",words)
        if len(item)>0:
            ret.append(item)
    return ret
df['Comment_token_punct'] = df.apply(lambda row :␣
↪post_punctuation(row['Comment_token']), axis = 1)
```

Stopwords such as "a," "the," and "and" do not provide additional information about a review and should be removed to reduce unnecessary computation and increase the emphasis on more important words.

```
#Stopwords
stop_words = set(stopwords.words('english'))
def remove_stopwords(x):
    filtered_sentence = []
    for w in x:
        if w not in stop_words:
            filtered_sentence.append(w)
    return filtered_sentence
df['Comment_token_punct_stopwords'] = df.apply(lambda␣
↪row : remove_stopwords(row['Comment_token_punct']),␣
↪axis = 1)
```

Here we will lemmatize our words so that the similar words such as fuel, fueling, and fueled will be converted to fuel. This is best done by providing the part of speech tags to the lemmatizer. The part of speech tag describes whether a word is used as a Noun, Adjective, Verb, or Adverb. Below we will use the averaged perceptron tagger to provide the part of speech tags for each token in the sentences.

```
#POS Tagging
nltk.download('averaged_perceptron_tagger')
df['pos_tags'] = df['Comment_token_punct_stopwords'].
↪apply(nltk.tag.pos_tag)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
```

(continues on next page)

(continued from previous page)

```
[nltk_data]    Package averaged_perceptron_tagger is already up-to-
[nltk_data]        date!
```

```
#wordnet POS - this gives us more accurate␣
↪lemmatization
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
df['wordnet_pos'] = df['pos_tags'].apply(lambda x:␣
↪[(word, get_wordnet_pos(pos_tag)) for (word, pos_
↪tag) in x])
#Lemmatization
wnl = WordNetLemmatizer()
df['lemmatized'] = df['wordnet_pos'].apply(lambda x:␣
↪[wnl.lemmatize(word, tag) for word, tag in x])
```

Now we will further remove duplicate words.

```
# ALl the duplicate words will be removed from the␣
↪text/comments including all the adjectives and␣
↪verbs.
def unique_(test_list):
    res = []
    for i in test_list:
        if i not in res:
            res.append(i)
    return res
df['Comment_token_punct_stopwords_unique'] = df.
↪apply(lambda row : unique_(row['lemmatized']), axis␣
↪= 1)
```

```
df.head()
```

```
   Counter                    Date              User  \
0        1   April 11, 2007 6:52PM     \r\nmotownusa
```

(continues on next page)

(continued from previous page)

```
1        2   April 11, 2007 7:33PM      \r\nexshoman
2        3   April 12, 2007 6:51AM  \r\ntargettuning
3        4   April 12, 2007 8:43AM           \r\npat
4        5  April 13, 2007 11:49AM         \r\nperna

                                                 Comment  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we're here...
4  \r\nThe Altima is my favorite of the bunch. It...

                                               Comment1  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI am sure some folks would appreciate havi...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we are her...
4  \r\nThe Altima is my favorite of the bunch. It...

                                          Comment_token  \
0  [hi, pat, :, you, forgot, the, chrysler, chrys...
1  [i, am, sure, some, folks, would, appreciate, ...
2  [you, can, try, to, revive, this, topic, but, ...
3  [model, vs., model, is, exactly, what, we, are...
4  [the, nissan, is, my, favorite, of, the, bunch...

                                       Comment_token_punct  \
0    [hi, pat, you, forgot, the, chrysler, chrysler]
1  [i, am, sure, some, folks, would, appreciate, ...
2  [you, can, try, to, revive, this, topic, but, ...
3  [model, vs, model, is, exactly, what, we, are,...
4  [the, nissan, is, my, favorite, of, the, bunch...

                             Comment_token_punct_stopwords  \
0               [hi, pat, forgot, chrysler, chrysler]
1  [sure, folks, would, appreciate, chevrolet, in...
2  [try, revive, topic, without, able, discuss, h...
3  [model, vs, model, exactly, manufacturer, vs, ...
4  [nissan, favorite, bunch, amongst, fastest, be...

                                                 pos_tags  \
0  [(hi, NN), (pat, NN), (forgot, VBD), (chrysler...
1  [(sure, JJ), (folks, NNS), (would, MD), (appre...
2  [(try, VB), (revive, JJ), (topic, NN), (withou...
3  [(model, NN), (vs, NN), (model, NN), (exactly,...
4  [(nissan, JJ), (favorite, JJ), (bunch, NN), (a...

                                              wordnet_pos  \
0  [(hi, n), (pat, n), (forgot, v), (chrysler, n)...
1  [(sure, a), (folks, n), (would, n), (appreciat...
2  [(try, v), (revive, a), (topic, n), (without, ...
3  [(model, n), (vs, n), (model, n), (exactly, r)...
4  [(nissan, a), (favorite, a), (bunch, n), (amon...

                                               lemmatized  \
0               [hi, pat, forget, chrysler, chrysler]
1  [sure, folk, would, appreciate, chevrolet, inc...
```

(continues on next page)

(continued from previous page)

```
2  [try, revive, topic, without, able, discus, ho...
3  [model, v, model, exactly, manufacturer, vs, m...
4  [nissan, favorite, bunch, amongst, fast, best,...

                Comment_token_punct_stopwords_unique
0                         [hi, pat, forget, chrysler]
1  [sure, folk, would, appreciate, chevrolet, inc...
2  [try, revive, topic, without, able, discus, ho...
3  [model, v, exactly, manufacturer, vs, belongs,...
4  [nissan, favorite, bunch, amongst, fast, best,...
```

For this exercise, we will focus on the top 10 most frequently discussed brands with sufficient data points.

```
#Frequency Distribution
from nltk.probability import FreqDist
fdist = FreqDist()
for i in range(len(df)):
    for word in df['Comment_token_punct_stopwords_
↪unique'][i]:
        fdist[word]+=1
```

```
freq_list = []
for i in range(len(model_details['brand'].unique())):
    if model_details['brand'].unique()[i] in fdist:
        l = model_details['brand'].unique()[i],␣
↪fdist[model_details['brand'].unique()[i]]
        freq_list.append(l)
freq_table = pd.DataFrame(freq_list,columns=["Model",
↪"Frequency"])
freq_table.sort_values(by = 'Frequency',ascending =␣
↪False, inplace = True, ignore_index=True)
```

```
freq_table.head(10)
```

```
       Model  Frequency
0      honda       2084
1       ford       1344
2     toyota        966
3    hyundai        598
4      mazda        576
5     nissan        564
6  chevrolet        246
7     saturn        245
```

(continues on next page)

(continued from previous page)

```
8    chrysler         221
9      subaru         158
```

```
col_req = 10
model_list = freq_table.head(col_req)['Model'].
↪tolist()
```

Now to see which brand is the closest competitor, we will see which brands are often mentioned within the same reviews. This will allow us to determine which cars are often being compared against one another and identify which is the closest competitor. We will create a co-occurrence matrix that counts the number of times two brands appear in a review.

```
cooccur_mat = pd.DataFrame(np.zeros((col_req,col_
↪req)), columns=model_list, index=model_list)
```

```
for rows in range(len(df)):
    for i in range(len(model_list)):
        if model_list[i] in df['Comment_token_punct_
↪stopwords_unique'][rows]:
            for j in range(i+1,len(model_list)):
                if model_list[j] in df['Comment_token_
↪punct_stopwords_unique'][rows]:
                    cooccur_mat[model_list[i]][model_
↪list[j]] = cooccur_mat[model_list[i]][model_
↪list[j]] + 1
                    cooccur_mat[model_list[j]][model_
↪list[i]] = cooccur_mat[model_list[i]][model_list[j]]
```

```
cooccur_mat
```

```
           honda   ford  toyota  hyundai  mazda  nissan  chevrolet  saturn  \
honda        0.0  652.0   690.0    358.0  275.0   339.0      137.0   131.0
ford       652.0    0.0   366.0    189.0  182.0   176.0       89.0    88.0
toyota     690.0  366.0     0.0    218.0  106.0   242.0       90.0   109.0
hyundai    358.0  189.0   218.0      0.0   75.0   108.0       52.0    54.0
mazda      275.0  182.0   106.0     75.0    0.0    94.0       29.0    31.0
nissan     339.0  176.0   242.0    108.0   94.0     0.0       55.0    77.0
chevrolet  137.0   89.0    90.0     52.0   29.0    55.0        0.0    70.0
saturn     131.0   88.0   109.0     54.0   31.0    77.0       70.0     0.0
chrysler   126.0   77.0    53.0     43.0   21.0    28.0       22.0    12.0
subaru      89.0   35.0    36.0     35.0   59.0    35.0       12.0    14.0

           chrysler  subaru
honda         126.0    89.0
ford           77.0    35.0
toyota         53.0    36.0
```

(continues on next page)

(continued from previous page)

```
hyundai        43.0    35.0
mazda          21.0    59.0
nissan         28.0    35.0
chevrolet      22.0    12.0
saturn         12.0    14.0
chrysler        0.0     7.0
subaru          7.0     0.0
```

Great, now we have created our co-occurrence matrix. However, this needs to be further processed as brands which are more talked about would have higher counts. In order to represent whether the comparison between two cars is more than normal, we will calculate the lift of two car brands. A lift greater than 1 indicates that the two brands are compared more than normal, whereas a lift less than 1 indicates the brands are compared less than normal.

```
len(df)
```

```
5000
```

```
lift = pd.DataFrame(np.zeros((col_req,col_req)),␣
↪columns=model_list, index=model_list)
```

```
total = len(df)
for i in range(col_req):
    for j in range(col_req):
        lift[model_list[i]][model_list[j]] =␣
↪total*(cooccur_mat[model_list[i]][model_list[j]]/
↪(freq_table.iloc[i,1]*freq_table.iloc[j,1]))
lift
```

```
              honda      ford    toyota   hyundai     mazda    nissan  \
honda      0.000000  1.163913  1.713737  1.436330  1.145467  1.442092
ford       1.163913  0.000000  1.409531  1.175794  1.175492  1.160925
toyota     1.713737  1.409531  0.000000  1.886897  0.952525  2.220901
hyundai    1.436330  1.175794  1.886897  0.000000  1.088698  1.601082
mazda      1.145467  1.175492  0.952525  1.088698  0.000000  1.446759
nissan     1.442092  1.160925  2.220901  1.601082  1.446759  0.000000
chevrolet  1.336158  1.345940  1.893652  1.767409  1.023318  1.982068
saturn     1.282855  1.336249  2.302784  1.842878  1.098356  2.786221
chrysler   1.367888  1.296192  1.241299  1.626841  0.824849  1.123199
subaru     1.351466  0.824103  1.179338  1.852165  3.241473  1.963821

            chevrolet    saturn  chrysler    subaru
honda        1.336158  1.282855  1.367888  1.351466
ford         1.345940  1.336249  1.296192  0.824103
toyota       1.893652  2.302784  1.241299  1.179338
hyundai      1.767409  1.842878  1.626841  1.852165
mazda        1.023318  1.098356  0.824849  3.241473
nissan       1.982068  2.786221  1.123199  1.963821
chevrolet    0.000000  5.807201  2.023323  1.543686
saturn       5.807201  0.000000  1.108136  1.808318
```

(continues on next page)

(continued from previous page)

```
chrysler    2.023323  1.108136  0.000000  1.002348
subaru      1.543686  1.808318  1.002348  0.000000
```

Now we will generate a dissimilarity matrix. To do this, we need to calculate the reciprocal of this function and replace infinity values with zero. This is because brands with higher lift values should be closer to one another than those with lower lift values in a distance metric. For example, the high lift value between the Mazda and Subaru should mean that the distance between them is smaller.

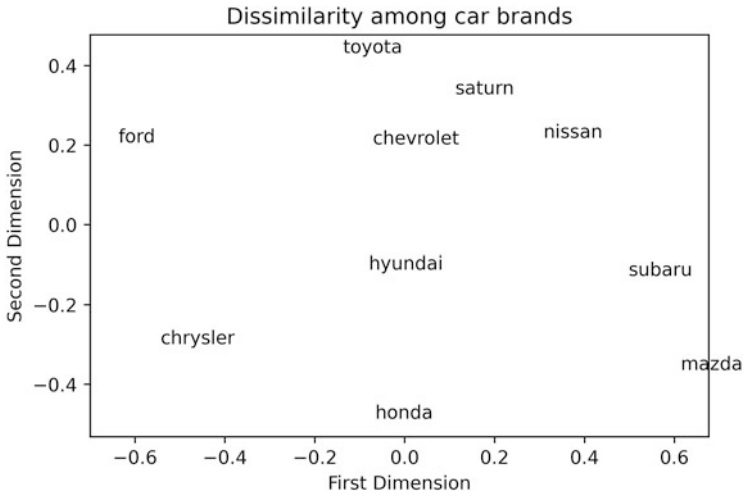
```
test = 1/lift
test = test.replace(np.inf, 0)
test
```

```
              honda      ford    toyota   hyundai     mazda    nissan  \
honda      0.000000  0.859171  0.583520  0.696219  0.873007  0.693437
ford       0.859171  0.000000  0.709456  0.850489  0.850708  0.861382
toyota     0.583520  0.709456  0.000000  0.529971  1.049842  0.450268
hyundai    0.696219  0.850489  0.529971  0.000000  0.918528  0.624578
mazda      0.873007  0.850708  1.049842  0.918528  0.000000  0.691200
nissan     0.693437  0.861382  0.450268  0.624578  0.691200  0.000000
chevrolet  0.748415  0.742975  0.528080  0.565800  0.977214  0.504524
saturn     0.779511  0.748364  0.434257  0.542630  0.910452  0.358909
chrysler   0.731054  0.771491  0.805608  0.614688  1.212343  0.890314
subaru     0.739937  1.213440  0.847933  0.539909  0.308502  0.509211

           chevrolet    saturn  chrysler    subaru
honda       0.748415  0.779511  0.731054  0.739937
ford        0.742975  0.748364  0.771491  1.213440
toyota      0.528080  0.434257  0.805608  0.847933
hyundai     0.565800  0.542630  0.614688  0.539909
mazda       0.977214  0.910452  1.212343  0.308502
nissan      0.504524  0.358909  0.890314  0.509211
chevrolet   0.000000  0.172200  0.494236  0.647800
saturn      0.172200  0.000000  0.902417  0.553000
chrysler    0.494236  0.902417  0.000000  0.997657
subaru      0.647800  0.553000  0.997657  0.000000
```

Now we will plot out our results by performing Multidimensional Scaling on our dataset. Multidimensional Scaling finds a nonlinear mapping to map the dissimilarity matrix provided by us, into a 2-dimensional coordinate system that preserves the dissimilarity. This allows us to intuitively visualize which brand is more similar to one another.

```
#Need to replace 'inf' with zero, otherwise the MDS␣
↪function does not work
from sklearn.manifold import MDS

mds = MDS(n_components=2, dissimilarity="precomputed",
↪ random_state=42)
results = mds.fit(test)
mds_coords = results.fit_transform(test)
```

```
#ANOTHER WAY TO PLOT THE MDS
coords = results.embedding_
plt.figure()
plt.scatter(mds_coords[:,0],mds_coords[:,1],
    facecolors = 'none', edgecolors = 'none')  #␣
↪points in white (invisible)
labels = freq_table['Model']
for label, x, y in zip(labels, mds_coords[:,0], mds_
↪coords[:,1]):
    plt.annotate(label, (x,y), xycoords = 'data')
plt.xlabel('First Dimension')
plt.ylabel('Second Dimension')
plt.title('Dissimilarity among car brands')
plt.show()
```




Now we have a nice visualization of how similar the public perception of various mid-sized sedan car brands is. From the graph, we can therefore extract valuable insights. Here are some example insights that we can infer from the graph:

1. We can see that Toyota is very close to Saturn and Chevrolet, which means that these are their closest competitors that they may need to watch out for as they are being compared more often than normal.
2. On the other hand, Toyota is rarely compared with Honda and Mazda. Therefore, competition against brands is less of a concern.
3. Interestingly, ford is significantly far from all other brands which indicates that they have been successful in brand differentiation.

### *16.3.3 SWOT Analysis*

SWOT analysis is a useful tool for understanding a company's current position and identifying potential opportunities and risks.

Once a SWOT analysis has been conducted, the information can be used to identify potential opportunities for the company to pursue and areas that need to be addressed. For example, if a company's strengths include a strong brand and a loyal customer base, it may want to consider a differentiation strategy that leverages those assets. On the other hand, if a company's weaknesses include high production costs, it may want to focus on a cost leadership strategy.

In addition, SWOT analysis can help a company to identify potential threats and take steps to mitigate them. For example, if a company identifies a potential new competitor entering the market, it may want to consider a strategy to defend its market position.

The results can then be used to inform the development of a business strategy that is well aligned with the company's strengths and weaknesses, as well as the external market opportunities and threats.

In this example, we will perform an automated SWOT analysis using reviews gathered from the forum in our analysis of the competitive landscape.

First let us reload the earlier dataframe as we will be breaking down the reviews into smaller sentences.

```
df = pd.read_csv(r"Business_Strategy\Text-Mining-Car-
↪Review-Data\edmunds_extraction.csv")
df.head()
```

```
   Counter                    Date                User  \
0        1   April 11, 2007 6:52PM     \r\nmotownusa
1        2   April 11, 2007 7:33PM      \r\nexshoman
2        3   April 12, 2007 6:51AM  \r\ntargettuning
3        4   April 12, 2007 8:43AM           \r\npat
4        5  April 13, 2007 11:49AM         \r\nperna

                                              Comment
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nModel vs. model is exactly what we're here...
4  \r\nThe Altima is my favorite of the bunch. It...
```

Once again we will perform decontraction

```
df['Comment1'] = df.apply(lambda row :␣
↪decontracted(row['Comment']), axis = 1)
df
```

```
      Counter                          Date                User  \
0           1       April 11, 2007 6:52PM      \r\nmotownusa
1           2       April 11, 2007 7:33PM       \r\nexshoman
2           3       April 12, 2007 6:51AM  \r\ntargettuning
3           4       April 12, 2007 8:43AM           \r\npat
4           5      April 13, 2007 11:49AM         \r\nperna
...       ...                         ...                ...
4995     4996  September 12, 2007 12:49PM      \r\nkdshapiro
4996     4997  September 12, 2007 12:55PM  \r\nbenderofbows
4997     4998   September 12, 2007 1:24PM      \r\njimlockey
4998     4999   September 12, 2007 2:21PM       \r\ncaptain2
4999     5000   September 12, 2007 2:24PM   \r\nthegraduate

                                                  Comment  \
0          \r\nHi Pat:You forgot the Chrysler Sebring
1     \r\nI'm sure some folks would appreciate havin...
2     \r\nYou can try to revive this topic but witho...
3     \r\nModel vs. model is exactly what we're here...
4     \r\nThe Altima is my favorite of the bunch. It...
...                                                 ...
4995  \r\n"Let me try one more time. Accord and Camr...
4996  \r\n"No one likes repairs on their car but I f...
4997  \r\nToyota & GM gives you guys what they want ...
4998  \r\nWhat weaknesses?  you have goota be kiddin...
4999  \r\nWell the cheapest service didn't have a pr...

                                                 Comment1
0          \r\nHi Pat:You forgot the Chrysler Sebring
1     \r\nI am sure some folks would appreciate havi...
2     \r\nYou can try to revive this topic but witho...
3     \r\nModel vs. model is exactly what we are her...
4     \r\nThe Altima is my favorite of the bunch. It...
...                                                 ...
4995  \r\n"Let me try one more time. Accord and Camr...
4996  \r\n"No one likes repairs on their car but I f...
4997  \r\nToyota & GM gives you guys what they want ...
4998  \r\nWhat weaknesses?  you have goota be kiddin...
4999  \r\nWell the cheapest service did not have a p...

[5000 rows x 5 columns]
```

Now we will perform sentence tokenization to break a long review into smaller sentences.

```
from nltk.tokenize import sent_tokenize

sent_df = {i:[] for i in df.columns}
sent_df["Sentence"] = []
for idx, line in df.iterrows():
    for sent in sent_tokenize(line["Comment1"].strip(
↪"\r\n. ")):
        for col in df.columns:
            sent_df[col].append(line[col])
```

(continues on next page)

(continued from previous page)

```
        sent_df["Sentence"].append(sent)
sent_df = pd.DataFrame.from_dict(sent_df)
```

```
sent_df.head()
```

```
   Counter                    Date                 User  \
0        1  April 11, 2007 6:52PM      \r\nmotownusa
1        2  April 11, 2007 7:33PM       \r\nexshoman
2        3  April 12, 2007 6:51AM   \r\ntargettuning
3        3  April 12, 2007 6:51AM   \r\ntargettuning
4        4  April 12, 2007 8:43AM            \r\npat

                                              Comment  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI'm sure some folks would appreciate havin...
2  \r\nYou can try to revive this topic but witho...
3  \r\nYou can try to revive this topic but witho...
4  \r\nModel vs. model is exactly what we're here...

                                             Comment1  \
0       \r\nHi Pat:You forgot the Chrysler Sebring
1  \r\nI am sure some folks would appreciate havi...
2  \r\nYou can try to revive this topic but witho...
3  \r\nYou can try to revive this topic but witho...
4  \r\nModel vs. model is exactly what we are her...

                                             Sentence
0             Hi Pat:You forgot the Chrysler Sebring
1  I am sure some folks would appreciate having t...
2  You can try to revive this topic but without b...
3  I do agree about issues with other members/pos...
4   Model vs. model is exactly what we are here for!
```

Now we will perform the same preprocessing on the sentences. However, we will also record down which brands were mentioned and what topics were the reviewers talking about. This will allow us to break down and understand which aspects of a brand are being talked about in the reviews.

```
sent_df['Sentence_token'] = sent_df.apply(lambda row␣
↪: token_(row['Sentence']), axis = 1)
sent_df['Sentence_token'] = sent_df.apply(lambda row␣
↪: lower_case(row['Sentence_token']), axis = 1)

model_details['brand'] = model_details.apply(lambda␣
↪row : lower(row['brand']), axis = 1)
model_details['model'] = model_details.apply(lambda␣
↪row : lower(row['model']), axis = 1)

sent_df['Brands'] = sent_df.apply(lambda row :␣
↪set([]), axis = 1)
```

(continues on next page)

(continued from previous page)

```
for i in range(len(sent_df)):
    for j in range(len(sent_df['Sentence_token'][i])):
        if sent_df['Sentence_token'][i][j] in model_
↪details['brand'].tolist():
            sent_df["Brands"][i].add(sent_df[
↪'Sentence_token'][i][j])

        if sent_df['Sentence_token'][i][j] in model_
↪details['model'].tolist():
            sent_df['Sentence_token'][i][j] = model_
↪details['brand'][model_details['model'].tolist().
↪index(sent_df['Sentence_token'][i][j])]
            sent_df["Brands"][i].add(sent_df[
↪'Sentence_token'][i][j])

#Replace certain related words to form larger␣
↪attribute categories
sent_df['Attributes'] = sent_df.apply(lambda row :␣
↪set([]), axis = 1)
for i in range(len(sent_df)):
    for j in range(len(sent_df['Sentence_token'][i])):
        if sent_df['Sentence_token'][i][j] in␣
↪attributes['word'].tolist():
            sent_df["Attributes"][i].add(attributes[
↪'category'][attributes['word'].tolist().index(sent_
↪df['Sentence_token'][i][j])])
            sent_df['Sentence_token'][i][j] =␣
↪attributes['category'][attributes['word'].tolist().
↪index(sent_df['Sentence_token'][i][j])]
```

Now we will drop all entries that do not contain any brands or attributes.

```
sent_df.drop(sent_df[sent_df["Brands"].apply(lambda␣
↪x: len(x)) == 0].index, inplace=True)
sent_df.drop(sent_df[sent_df["Attributes"].
↪apply(lambda x: len(x)) == 0].index, inplace=True)
len(sent_df)
```

```
2438
```

Now we will drop duplicate words.

```
sent_df['Sentence_token_punct_stopwords_unique'] =␣
↪sent_df.apply(lambda row : unique_(row['Sentence_
↪token']), axis = 1)
```

Merge the tokens together into complete sentences for our sentiment analyzer to take in.

```
sent_df['Processed_Sentence'] = sent_df.apply(lambda␣
↪row : ' '.join(row['Sentence_token_punct_stopwords_
↪unique']), axis = 1)
```

Now let us begin by importing Vader, our sentiment analysis model, and use it to score the sentiment of the sentences.

```
from nltk.sentiment.vader import␣
↪SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()
sent_df["vader_score"] = sent_df.apply(lambda row :␣
↪sid.polarity_scores(row['Processed_Sentence']),␣
↪axis = 1)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\YuJin\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

We will collate the results in separate rows.

```
sent_df['neutral'] = sent_df.apply(lambda row : row[
↪'vader_score']["neu"], axis = 1)
sent_df['sentiment'] = sent_df.apply(lambda row : row[
↪'vader_score']["compound"], axis = 1)
```

Let us drop all sentences that are completely neutral.

```
sent_df.drop(sent_df[sent_df['neutral'] == 1.0].index,
↪ inplace=True)
```

Now we want to find out which attributes were more discussed about for car brands.

```
discussed_attributes = {}
for attrs in sent_df["Attributes"]:
    for attr in attrs:
        if attr in discussed_attributes:
            discussed_attributes[attr] += 1
        else:
            discussed_attributes[attr] = 1

discussed_attributes = pd.DataFrame.from_
↪dict(discussed_attributes, orient="index", columns=[
↪"Frequency"])
discussed_attributes = discussed_attributes.reset_
↪index()
discussed_attributes.sort_values(by = 'Frequency',
↪ascending = False, inplace = True, ignore_
↪index=True)
attr_list = discussed_attributes['index'].tolist()
attr_list
```

```
['performance', 'look', 'cost', 'gas', 'size']
```

Now we can see the most discussed topics with engine having the highest frequency. Let us break our dataframe into smaller dataframes, specific to each of the topics identified.

```
topic_dfs = {}
for attr in attr_list:
    topic_dfs[attr] = sent_df[sent_df.apply(lambda␣
↪row: attr in row["Attributes"], axis = 1)]
```

Now that we have broken down our data into relevant topics, let us perform a SWOT analysis for Saturn.

In SWOT analysis, we can define the strengths of our company as areas where our company has a higher sentiment score in comparison to the industry average. Similarly, the weaknesses will be defined as area which our company is performing poorly in as compared to the industry.

Opportunities and threats are external factors which will come from our competitive landscape. This will be done by analyzing the sentiments of our competitors and categorizing them as threats if they are close or above our sentiment score for that category. On the other hand, if the sentiments of our competitors are lower compared to ours significantly, we will classify it as an opportunity that we can differentiate our brand by and capitalize on.

To make things simpler, let us analyze reviews which mention about 1 brand in the sentence as comparisons between two brands can be more complex to analyze.

First let us begin by defining the function that will provide us with brand and industry scores.

```
def get_brand_sentiment_scores(brand, attr_list,␣
↪topic_dfs):
    brand_scores = {}
    industry_scores = {}
    for attr in attr_list:
        topic_df = topic_dfs[attr]
        industry_score = topic_dfs[attr]["sentiment"].
↪mean()
        brand_topic_df = topic_df[topic_df.
↪apply(lambda row: (brand in row["Brands"]) and␣
↪(len(row["Brands"]) == 1), axis = 1)]
        brand_score = brand_topic_df["sentiment"].
↪mean()
        if np.isnan(brand_score):
            # For attributes which weren't mentioned␣
↪in the reviews, we will set it equal to the industry
            brand_scores[attr] = industry_score
        else:
            brand_scores[attr] = brand_score
        industry_scores[attr] = industry_score
    return brand_scores, industry_scores
```

Great, now let us pretend that we are working for Chevrolet and want to perform a SWOT analysis for our company. Let us see how we fare.

```
brand = "chevrolet"
chevrolet_scores, industry_scores = get_brand_
↪sentiment_scores(brand, attr_list, topic_dfs)

for attr, score in chevrolet_scores.items():
    print(attr, round(score, 3), round(industry_
↪scores[attr], 3))
```

```
performance 0.108 0.229
look 0.469 0.33
cost 0.639 0.227
gas 0.219 0.229
size 0.178 0.296
```

Our scores look pretty decent.

Now that we have our brand's sentiment scores against the larger automotive industry, let us categorize them into strengths and weaknesses.

Let us set some thresholds to define what would we consider a strength or weakness. Here we will define attributed with scores that are at least 0.05 higher against the industry to be classified as strengths, and attributes with scores at least 0.05 below the industry to be considered as weaknesses.

```
Strengths = []
Weaknesses = []

for attr, score in chevrolet_scores.items():
    if (score - industry_scores[attr]) > 0.05:
        Strengths.append("Our " + attr + " is better␣
↪than the industry standard, we can use this to our␣
↪advantage")
    if (score - industry_scores[attr]) < -0.05:
        Weaknesses.append("Our " + attr + " is worse␣
↪than the industry standard, we will need to improve␣
↪in this aspect")
```

Now let us see what the scores are for some of our closest competitors, Saturn and Toyota.

```
brand = "toyota"
toyota_scores, industry_scores = get_brand_sentiment_
↪scores(brand, attr_list, topic_dfs)

for attr, score in toyota_scores.items():
    print(attr, round(score, 3), round(industry_
↪scores[attr], 3))
```

```
performance 0.231 0.229
look 0.446 0.33
cost 0.127 0.227
gas 0.22 0.229
size 0.3 0.296
```

```
brand = "saturn"
saturn_scores, industry_scores = get_brand_sentiment_
↪scores(brand, attr_list, topic_dfs)

for attr, score in saturn_scores.items():
    print(attr, round(score, 3), round(industry_
↪scores[attr], 3) )
```

```
performance 0.294 0.229
look 0.305 0.33
cost 0.272 0.227
gas 0.229 0.229
size 0.129 0.296
```

We will now compare Chevrolet's sentiment scores to our competitors' sentiment scores. This time we will be stricter as it is important to monitor companies that we are close to in terms of performance. Therefore, we will list competitors that are within 0.05 of our score to be threats that need to be monitored. Subsequently, we will list our competitors' attributes that have a score greater than our corresponding score $+0.05$ as threats that are stronger than us. Lastly, we will list our competitors' attributes that have a score less than our corresponding score $-0.05$ as opportunities.

```
Opportunities = []
Threats = []

for attr, score in chevrolet_scores.items():
    if toyota_scores[attr] < score - 0.05:
        Opportunities.append("Toyota's " + attr + "␣
↪is weaker than our's, we can leverage on this␣
↪opportunity")
    if abs(toyota_scores[attr] - score) < 0.05:
        Threats.append("Toyota's " + attr + " is␣
↪close to our's, we will need to monitor the␣
↪situation carefully")
    elif toyota_scores[attr] > score - 0.05:
        Threats.append("Toyota's " + attr + " is␣
↪stronger than our's, this is a concern we need to␣
↪address")

for attr, score in chevrolet_scores.items():
    if saturn_scores[attr] < score - 0.05:
        Opportunities.append("Saturn's " + attr + "␣
↪is weaker than our's, we can leverage on this␣
↪opportunity")
    if abs(saturn_scores[attr] - score) < 0.05:
        Threats.append("Saturn's " + attr + " is␣
↪close to our's, we will need to monitor the␣
↪situation carefully")
    elif saturn_scores[attr] > score - 0.05:
```

(continues on next page)

(continued from previous page)

```
        Threats.append("Saturn's " + attr + " is␣
↪stronger than our's, this is a concern we need to␣
↪address")
```

Let us collate our results from our SWOT analysis.

```
print("Strengths: ")
for strength in Strengths:
    print(strength)
print()

print("Weaknesses:")
for weakness in Weaknesses:
    print(weakness)
print()

print("Opportunities: ")
for opportunity in Opportunities:
    print(opportunity)
print()

print("Threats:")
for threat in Threats:
    print(threat)
```

```
Strengths:
Our look is better than the industry standard, we can␣
↪use this to our advantage
Our cost is better than the industry standard, we can␣
↪use this to our advantage

Weaknesses:
Our performance is worse than the industry standard,␣
↪we will need to improve in this aspect
Our size is worse than the industry standard, we will␣
↪need to improve in this aspect

Opportunities:
Toyota's cost is weaker than our's, we can leverage␣
↪on this opportunity
Saturn's look is weaker than our's, we can leverage␣
↪on this opportunity
```

(continues on next page)

(continued from previous page)

```
Saturn's cost is weaker than our's, we can leverage␣
 ↪on this opportunity

Threats:
Toyota's performance is stronger than our's, this is␣
 ↪a concern we need to address
Toyota's look is close to our's, we will need to␣
 ↪monitor the situation carefully
Toyota's gas is close to our's, we will need to␣
 ↪monitor the situation carefully
Toyota's size is stronger than our's, this is a␣
 ↪concern we need to address
Saturn's performance is stronger than our's, this is␣
 ↪a concern we need to address
Saturn's gas is close to our's, we will need to␣
 ↪monitor the situation carefully
Saturn's size is close to our's, we will need to␣
 ↪monitor the situation carefully
```

Great, so now we know that we are maintaining our competitiveness against Toyota and Saturn through our cost advantage and better looking cars. On the other hand, we will need to improve our performance and size.

## Exercises

1. Define what is a business strategy.
2. List three different examples of business strategies.
3. List three different commonly used business strategy frameworks and explain how it helps the business in developing a business strategy.
4. List three different examples of barriers to entry.
5. Identify two ways that artificial intelligence can be applied to business strategy.

## References

1. Afranur (2020) Machine learning algorithms with PCA. https://www.kaggle.com/code/afranur/machine-learning-algorithms-with-pca. Accessed 27 Apr 2023
2. FasterCapital (2022) The business community's guide to acquiring a successful startup. https://fastercapital.com/content/The-Business-Community-s-Guide-to-Acquiring-a-Successful-Startup.html. Accessed 27 Apr 2023
3. Manishkc06 (2020) Startup success prediction. https://www.kaggle.com/datasets/manishkc06/startup-success-prediction. Accessed 27 Apr 2023

4. Shapiro D (2019) Can artificial intelligence generate corporate strategy? https://www.forbes.com/sites/danielshapiro1/2019/08/19/can-artificial-intelligence-generate-corporate-strategy/. Accessed 27 Apr 2023
5. Tarashjain (2021) Github—tarashjain/text-mining-car-review-data. https://github.com/tarashjain/Text-Mining-Car-Review-Data. Accessed 27 Apr 2023
6. VectorMine (2023) BCG Matrix vector illustration outlined cash stock vector (royalty free) 1341905255 | Shutterstock. https://www.shutterstock.com/image-vector/bcg-matrix-vector-illustration-outlined-cash-1341905255. Accessed 13 May 2023
7. VectorMine (2023) Free swot vector illustration—vectormine. https://vectormine.com/item/free-swot-scheme-outline-vector-illustration-icons/. Accessed 13 May 2023
8. VectorMine (2023) Pestle vector illustration labeled market cognition stock vector (royalty free) 1329752819 | Shutterstock. https://www.shutterstock.com/image-vector/pestle-vector-illustration-labeled-market-cognition-1329752819. Accessed 13 May 2023
9. VectorMine (2023) Product life cycle vector illustration outlined stock vector (royalty free) 1344410870 | Shutterstock. https://www.shutterstock.com/image-vector/product-life-cycle-vector-illustration-outlined-1344410870. Accessed 13 May 2023
10. VectorMine (2023) Value chain vector illustration outlined product stock vector (royalty free) 1341923684 | Shutterstock. https://www.shutterstock.com/image-vector/value-chain-vector-illustration-outlined-product-1341923684. Accessed 13 May 2023

# Index





T. T. Teoh, Y. J. Goh, *Artificial Intelligence in Business Management*,
Machine Learning: Foundations, Methodologies, and Applications,
https://doi.org/10.1007/978-981-99-4558-0