

# Microsoft Azure AI A Beginner's Guide

Explore Azure Applied AI Services, Azure Cognitive Services  
and Azure Machine Learning with Practical Illustrations

REKHA KODALI  
SANKARA NARAYANAN GOVINDARAJULU  
MOHAMMED ATHAULLA






# Microsoft Azure AI

## A Beginner's Guide

Explore Azure Applied AI Services, Azure Cognitive Services  
and Azure Machine Learning with Practical Illustrations



REKHA KODALI

SANKARA NARAYANAN GOVINDARAJULU

MOHAMMED ATHAULLA



# **Microsoft Azure AI: A Beginner's Guide**

---

*Explore Azure Applied AI Services,  
Azure Cognitive Services and Azure Machine  
Learning with Practical Illustrations*

---

**Rekha Kodali  
Sankara Narayanan Govindarajulu  
Mohammed Athaulla**



[www.bpbonline.com](http://www.bpbonline.com)

**FIRST EDITION 2022**

**Copyright © BPB Publications, India**

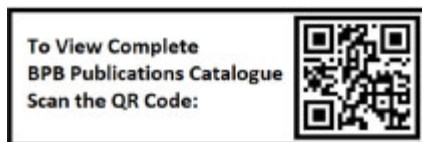
**ISBN: 978-93-55510-518**

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

#### **LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY**

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.



[www.bpbonline.com](http://www.bpbonline.com)

# Foreword

As **Senior Vice President – Everything on Azure Solutions at Avanade Inc**, I see Artificial Intelligence (AI) as the core pillar for Digital Transformation across all verticals and customer segments.

The concept of AI has fascinated a lot of free thinkers and frightened others. A few refer to AI as a great leap towards building future businesses, while some bashes the idea based on fears that might lead to human extinction.

While AI technologies are still far from creating a self-aware and self-evolving AI that might take over the earth, for a fact - AI already has a big impact on almost every facet of our lives. AI has given businesses across industries a chance to leverage it to build future products, services, optimize operations, and drive invaluable insights for improving Customer experience and Employee experience. Each year, new developments are being made with artificial intelligence and automation that businesses can use in various innovative ways. **But guess what, the availability of AI Skilled Professionals is one biggest blocker in the accelerated adoption of AI.**

What like about this book is that it provides practical guidelines for different personas right from CTO level to AI Architect or AI Consultant or a beginner AI Developer. This book makes it easy to learn Microsoft Azure AI concepts, common use cases, covers the basics and advanced concepts of Azure AI. The book is well structured and covers a vast number of Azure AI and PaaS capabilities. The best way to learn new technology is to try your hands on developing simple apps and writing the code. This book helps with addressing that aspect with well-explained working code samples.

I enjoyed the chapter covering the basics of the Azure AI platform, and concepts of the Azure AI ecosystem and services. This is a must-read book if you want to learn about Azure Cognitive Services, Azure Computer Vision, Azure Applied AI Services, and Azure Machine Learning with practical examples to build intelligent applications in the domain of Image

Processing, Object Detection, Text Recognition, OCR, Spatial Analysis and Face Recognition.

Do not forget to test your knowledge against well-crafted Multiple-Choice questions at the end of each chapter. Nice to reinforce the learning objective.

Book also provides technical content and insights Azure Applied AI Services like Azure Form Recognizer, Azure Metrics Advisor, Azure Cognitive Search, Azure Immersive Reader, Azure Video Analyzer for Media, and Azure Bot Service.

Books do a great job in developing a good understanding of BoT development options, BOT Framework, how to leverage Bot Framework Emulator, NLP capabilities to enhance conversational interfaces.

From me bonus, part of the book is the concluding chapter, which covers concepts about infusing ML in Custom Applications using ML.NET.

Perhaps the most remarkable thing about this book is that it provides an AI practitioner perspective and structured approach to learning Azure AI.

By the end of this book, I am sure, you would have immersed yourself with practical experience of working with Azure AI, Azure ML services, and APIs and tools available in the Azure AI Platform.

Wishing the authors all the very best for this new endeavor.



Gaurav Aggarwal

# Dedicated to

*My Parents*

***Dr. Vaikunta Rao***

*&*

***Ramamani***

—*Rekha*



# About the Authors

**Rekha Kodali** is a Principal Director in Accenture. With more than 23 years of experience in Microsoft Technologies, her focus areas include Azure, Enterprise Architecture, Microservices, Solution Architecture, Presales and Innovation. She has been instrumental in creating differentiated solutions and service offerings. Her certifications include TOGAF, IASA certified Foundation Architect and Azure Solutions Architect Expert and several other MS certifications. She has published multiple research papers.

She is the author of a book: **Developing Cloud Native Applications In Azure Using .Net Core: A Practitioner's Guide to Design, Develop and Deploy Apps** Paperback – 1.

LinkedIn Profile: <https://www.linkedin.com/in/rekha-kodali-3561201b/>

**Sankara Narayanan Govindarajulu** is a Solution Architect at Microsoft, helping with partners and customers on their digital transformation journey. Throughout his career of 23 years, Sankara has put his profound experience along with his passion in compelling ways to help customers in strategic decision making, project execution, architecture and design, and development of enterprise solutions covering various aspects such as availability, performance, security, business continuity, cloud migration and transformation across platforms and technologies. Sankara is TOGAF certified, and has certifications in Azure Migration, Architecture, Development and Security. He has published a book, a few journals and presented in internal technical conferences on multiple occasions.

His LinkedIn Profile: <https://www.linkedin.com/in/sankara-narayanan-g/>

**Mohammed Athaulla** is a Lead developer at Wipro Technologies with more than 10 years of experience in Microsoft technologies. He has hands-on experience in development and customization of software applications using various technologies such as Azure, ASP.net core MVC, Web API,



Angular, JavaScript, SQL Server etc. He led the development of a large AI based solution.

His LinkedIn Profile is <https://www.linkedin.com/in/athaulla/>

# About the Reviewer

**Suryanarayana Murty Eranki** has 24 years of practice leadership in architecting & delivering cloud solutions, with significant experience in IT software. He is an accomplished and technically sophisticated leader with proven success in architecting and implementing large, complex, and business critical technology projects across the project life cycle. As a Cloud solutioning expert, helping enterprises adopt public cloud platforms such as Microsoft Azure, Amazon Web Services and Google. He has worked with companies like Wipro, Tech Mahindra, Cognizant, Genpact and currently he is leading a Cloud consulting boutique, Apware - Hyderabad as an Executive Director to deliver Microsoft Azure and AI/ML focussed solutions to their customers.

LinkedIn Profile: [\*\*https://www.linkedin.com/in/eranki/\*\*](https://www.linkedin.com/in/eranki/)

# Acknowledgements

**Rekha Kodali:** I would like to thank and acknowledge the guidance, inspiration and support provided by my mentors Venkata Guru Prasad Kandarp and Hari Kishan Burle. I would like to thank my kids (Sireesh and Tarun), my brothers and Sarath, my husband for their support.

**Sankara Narayanan Govindarajulu:** Thankful to family, friends, colleagues and reviewers whose support and inspiration made this possible

**Mohammed Athaulla:** I am extremely grateful to my parents, wife, and son (Zayan). It would not have been possible without their support.

# Preface

The recent years have proved that AI is quickly becoming the “new normal” at organizations of all sizes. Organizations across industries are revamping their strategies and investments for future with AI becoming one of the top priorities. The objective of this book is to provide practical guidelines that might help you if you are in a role of a CTO/ Architect / Designer / Consultant / Developer or even a Beginner in their AI journey. We believe this book will make it easy to learn Azure AI concepts and covers the basics and also some advanced concepts of Azure AI. The book is divided into 8 chapters. Software professionals interested in Azure AI development would benefit from this book.

[Chapter 1](#) gives a basic introduction to the services/components provided by Azure to build AI based applications, Azure Cognitive Services, and Azure Computer Vision. After reading this chapter, the reader will be able to understand Azure AI Platform, the core concepts of Azure Cognitive Services and various services available within Azure Computer Vision.

[Chapter 2](#) introduces Azure Vision APIs for developing custom applications which can be used for processing images, recognizing faces and detecting objects. After reading this chapter, the reader will have the skills that make it possible to effectively build applications by leveraging Image Analytics, Content Moderation, Object Detection, Face Recognition, and Custom Vision Services.

[Chapter 3](#) introduces Handwriting Recognition, Optical Character Recognition, Face API, and Spatial Analysis. After reading this chapter, the reader will have the skills that make it possible to effectively build applications by leveraging Handwriting Recognition, Optical Character Recognition, Face API and Spatial Analysis.

[Chapter 4](#) explores how to leverage Decision, Language, Speech, and Web Search in the applications. After reading this chapter, the reader will understand how to effectively build applications leveraging each of the APIs.

**Chapter 5** makes the reader understand Azure Applied AI Services. We will delve a little deep into Azure Form Recognizer, Azure Metrics Advisor, Azure Cognitive Search, Azure Immersive Reader, and Azure Video Analyzer for Media. By the end of this chapter, the reader will understand how to effectively leverage these services.

**Chapter 6** takes the reader through the Microsoft Bot Landscape including the Bot Framework, the Bot Framework Composer after a brief introduction of Bots and how they have evolved over the years. We will look at what the different options have to offer Bot Framework has to offer, the best practices and scenarios, a simple example using the Bot Framework with using QnA and include NLP (LUIS) capabilities as well and also understand how to develop/debug Bots locally with the Bot Emulator, Bot Composer and its features and a quick example.

**Chapter 7** introduces ML.NET. The reader will learn how to leverage ML.NET which is a cross-platform, open source machine learning framework. After reading this chapter the reader will understand how a ML model can be developed and then exposed as an API or a web service. The reader will also understand how it can be invoked from .NET code.

**Chapter 8** introduces Azure Machine Learning Studio. The reader will learn to create models, score models and run experiments and also learn about Accord Framework. The reader will get a view on ML Ops and a brief overview of migration from classic ML Studio Azure Designer.

# Code Bundle and Coloured Images

Please follow the link to download the  
*Code Bundle* and the *Coloured Images* of the book:

**<https://rebrand.ly/137e75>**

The code bundle for the book is also hosted on GitHub at <https://github.com/bpbpublications/Microsoft-Azure-AI-A-Beginner-s-Guide>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at <https://github.com/bpbpublications>. Check them out!

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**[errata@bpbonline.com](mailto:errata@bpbonline.com)**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.bpbonline.com](http://www.bpbonline.com) and as a print book

customer, you are entitled to a discount on the eBook copy. Get in touch with us at: [business@bpbonline.com](mailto:business@bpbonline.com) for more details.

At [www.bpbonline.com](http://www.bpbonline.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.



## **Piracy**

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [business@bpbonline.com](mailto:business@bpbonline.com) with a link to the material.

## **If you are interested in becoming an author**

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit [www.bpbonline.com](http://www.bpbonline.com). We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

## **Reviews**

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit [www.bpbonline.com](http://www.bpbonline.com).

# Table of Contents

## 1. Azure AI Platform and Services

Structure

Objective

Azure AI platform

*Azure Applied AI Services*

AI Services

*Trained Services (Prebuilt)*

Conversational AI

Custom services

Azure Cognitive Services

Azure Vision Service

Accessing the APIs

Use cases

Setting up Developer Tools

Conclusion

Multiple choice questions

*Answers*

## 2. Azure Computer Vision - Image Analysis, Processing, Object Detection

Structure

Objective

Image processing

Image analytics - Azure Computer Vision API

Image Analytics -Lab

Image Classification using Azure Custom Vision Service

*Azure Custom Vision Service-Image Classification Lab*

Object detection using Azure Custom Vision Service

Conclusion

Multiple choice questions

*Answers*

### **3. Computer Vision - Optical Character Recognition, Face API, and Spatial Analysis**

Structure

Objective

Handwritten / Printed text recognition

Handwriting recognition lab

Face recognition

Computer Vision - Face API - In Detail

Computer Vision - Face API Lab

Spatial Analysis

Sample use cases

Conclusion

Multiple choice questions

Questions

Answers

### **4. Azure Cognitive Services**

Structure

Objective

Decision service

*Content Moderator-Use case*

*Content Moderator-Detailed APIs*

Personalizer

Language Service

Azure Logic Apps and Functions

Lab - Create Logic App using OCR and Text Analytics and Azure Functions

Speech

Bing Web Search

*Lab*

Azure Cognitive Services containers

Conclusion

Multiple Choice Questions

Questions

Answers

### **5. Azure Applied AI Services**

[Structure](#)  
[Objective](#)  
[Overview of Azure Applied AI Services](#)  
[Azure Metrics Advisor](#)  
[Azure Immersive Reader](#)  
[Azure Video Analyzer for Media](#)  
[Azure Form Recognizer](#)  
[Azure Cognitive Search](#)  
[Lab: Creating a Azure Cognitive Search Service](#)  
[Conclusion](#)  
[Multiple choice questions](#)  
[Answers](#)

## **6. Bots - A Brief Introduction**

[Structure](#)  
[Objective](#)  
[How Machines Learn – A Brief Introduction](#)  
[Introducing Bots](#)  
[The Bot Ecosystem](#)  
[\*The Bot Framework\*](#)  
[\*The Bot Framework Composer\*](#)  
[\*Cognitive Intelligence\*](#)  
[\*QnA Maker\*](#)  
[\*LUIS - Language Understanding Intelligent Service\*](#)  
[\*Azure Cognitive Services for Language\*](#)  
[\*Search\*](#)  
[\*Storage\*](#)  
[\*Information sources\*](#)  
[\*Build Bots using the Bot Framework\*](#)  
[Lifecycle](#)  
[Reference Architecture](#)  
[The Bot Framework Emulator](#)  
[\*Building a Basic Bot\*](#)  
[\*Creating a QnA KB\*](#)  
[\*Using the Emulator\*](#)  
[\*Adding LUIS to the sample\*](#)  
[\*Adding a LUIS Model\*](#)

[\*Adding a second QnA KB\*](#)  
[\*Creating a dispatch model using the Bot Framework Orchestrator\*](#)  
[\*Updating the BasicBot source code to include dispatch support\*](#)  
[\*Build and Test it using Emulator\*](#)  
[\*Adding the Oliver Garden FAQ as an exercise\*](#)  
[\*Enhancing it further\*](#)  
[Building the Bot Framework Composer](#)  
[\*Enterprise Scenarios\*](#)  
[\*A Customer scenario\*](#)  
[\*Typical use cases\*](#)  
[Conclusion](#)  
[Thought Experiment](#)

## **7. Infusing ML in Custom Applications Using ML.NET**

[Structure](#)  
[Objective](#)  
[Introducing ML.NET](#)  
[\*Algorithms supported by ML.NET\*](#)  
[\*Choosing the right algorithm\*](#)  
[\*Trainers\*](#)  
[Creating a ML.NET Model](#)  
[\*Samples\*](#)  
[Lab 1 - Creating Samples using ML.NET - Sentiment Analysis](#)  
[Lab 2 - Creating samples using ML.NET - Fare prediction](#)  
[Lab 3 - Creating samples using ML.NET - Issue classification](#)  
[\*ML.NET Model Builder\*](#)  
[Lab 4 - Creating sample using ML.NET Model Builder for sentiment analysis](#)  
[Conclusion](#)  
[Multiple choice questions](#)  
[\*Answers\*](#)

## **8. Using Azure ML Studio**

[Structure](#)  
[Objective](#)  
[Azure Machine Learning Studio](#)

[Sample lab using Azure ML Studio using Automated machine learning UI no-code approach](#)

[Sample Lab using Azure ML Studio Azure Machine Learning designer ML Ops](#)

[Migration](#)

[Azure ML library of algorithms](#)

[Accord.NET -A brief Introduction](#)

[Conclusion](#)

[Multiple choice questions](#)

[Answers](#)

**[Index](#)**

# **CHAPTER 1**

## **Azure AI Platform and Services**

This chapter gives a basic introduction to the services/components provided by Azure to build AI based applications, Azure Cognitive Services, and Azure Computer Vision. It gives an overview of the available APIs and then gives an introduction to some sample use cases possible with Azure Computer Vision.

### **Structure**

This chapter will cover the following topics:

- Platform provided by Azure to build AI-based applications
- Azure Cognitive Services – Core Aspects
- Conversational AI
- Azure Computer Vision
- Use Cases – Enterprise scenarios
- Setting up developer tools

### **Objective**

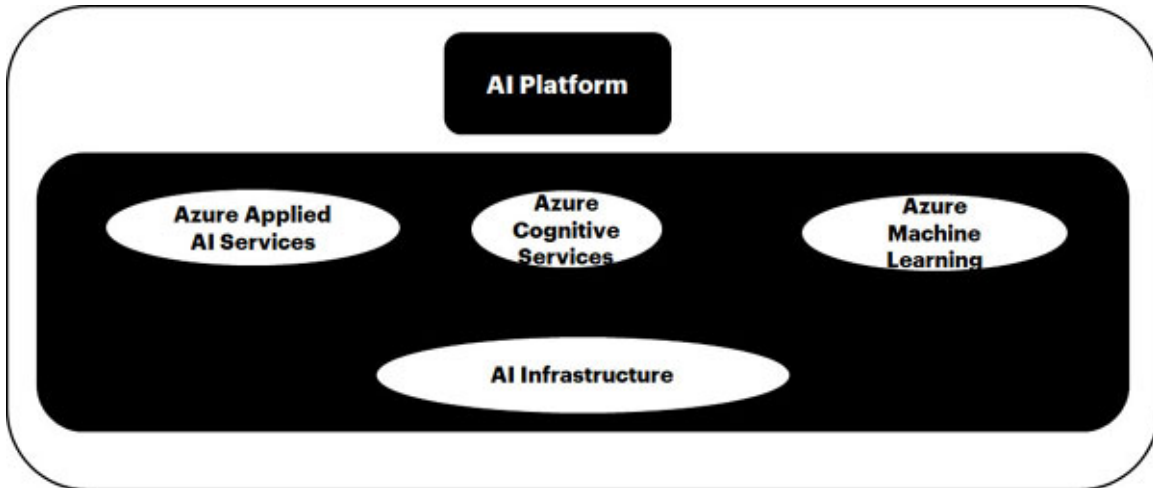
After reading this chapter, the reader will be able to understand the Azure AI platform, the core concepts of Azure Cognitive Services, various services available within Azure Computer Vision, and specific use cases leveraging Azure Computer Vision.

### **Azure AI platform**

Intelligent applications leverage Artificial Intelligence to deliver rich, adaptive, personalized, immersive, contextual experiences to users intelligently and are autonomous. Natural interfaces of text, speech, and vision unlocking new categories such as conversational commerce and AR/VR help in building immersive applications. Immersive applications incorporate intelligent features



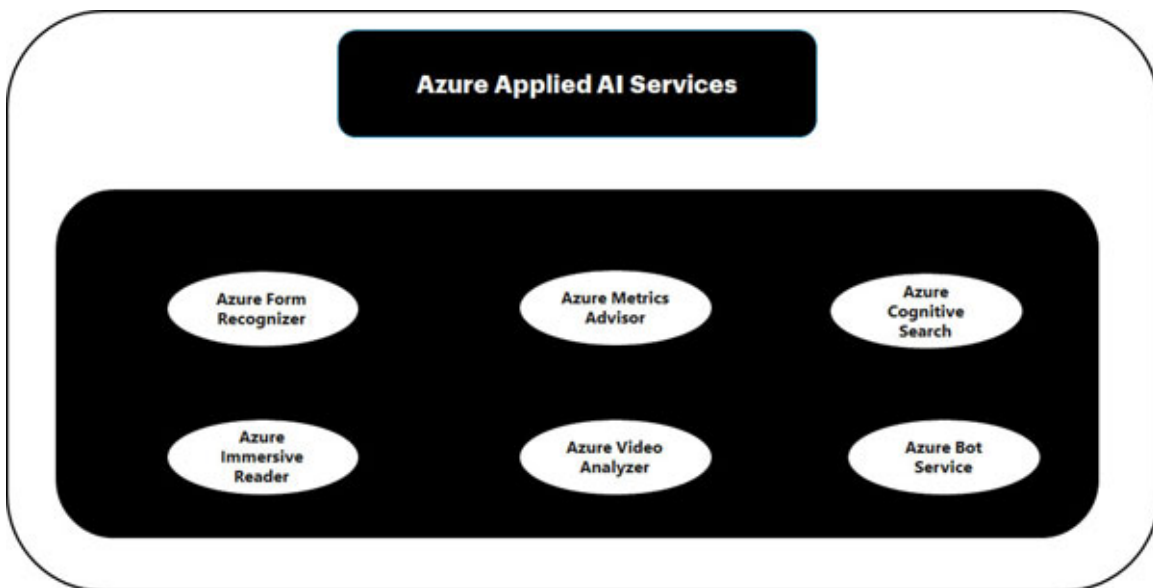
such as emotion and sentiment detection, vision and speech recognition, language understanding, knowledge, and search. Azure AI Platform provides the capability to build such applications very quickly. Azure AI Platform can be broadly classified into four Services/Components as depicted below:



*Figure 1.1: Components provided by Azure to build AI-based applications*

## [Azure Applied AI Services](#)

Azure Applied AI Services are built leveraging Azure Cognitive Services. They help in tasks like boosting literacy in the classroom, diagnosing, monitoring anomalies in metrics, knowledge from mining documents, document understanding, transcription analysis, and more scenarios.



*Figure 1.2: Azure Applied AI Services*

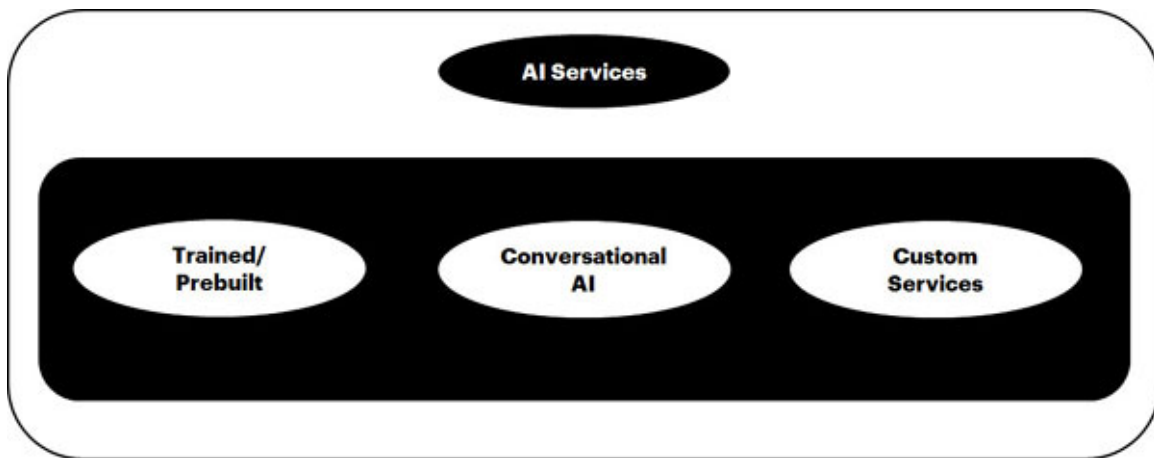
- **Azure Form Recognizer:** It helps in identifying and extracting text, key/value pairs, and table data from form documents. It helps in ingesting text from forms and outputs structured data that includes the relationships in the original file.
- **Azure Metrics Advisor:** It helps in identifying and fixing problems leveraging a combination of near-real-time monitoring, adapting models to specific scenarios and helps with diagnostics and alerts.
- **Azure Cognitive Search:** It leverages built-in AI capabilities that help in identifying and exploring relevant content.
- **Azure Immersive Reader:** It is an inclusively designed tool that implements techniques to improve reading comprehension for language learners, new readers, and people with learning differences.
- **Azure Video Analyser for Media:** It helps extract metadata such as spoken words, faces etc from video and audio files.
- **Azure Bot Service:** It helps in developing conversational AI experiences.
- **Azure Cognitive Services:** Developers can rapidly consume high-level “finished” services that accelerate the development of AI solutions. They help in composing intelligent applications, customized to the organization’s availability, security, and compliance requirements.
- **Azure Machine Learning:** Consists of a set of comprehensive tools and frameworks that help build, deploy, and operationalize ML models. They provide an extensive set of supported tools and IDEs and Azure ML studio that help in easily building and fine-tuning ML models and deploying them.
- **AI Infrastructure:** AI Services and tools are backed by providing access to large scale infrastructure. It provides hyper-clusters of thousands of state-of-the-art GPUs, a breadth of AI hardware that includes the most comprehensive set of GPUs, and an array of general-purpose CPUs. It provides the latest high-bandwidth networks inside of every server.

We will look at some of the different types of applications that can be built using the Azure AI platform.

## [AI Services](#)

AI services help access high-quality speech, vision, decision-making, language, and AI models through simple REST API calls and also provide the

ability to create our own machine learning models.  
AI services offered can be broadly classified as follows:



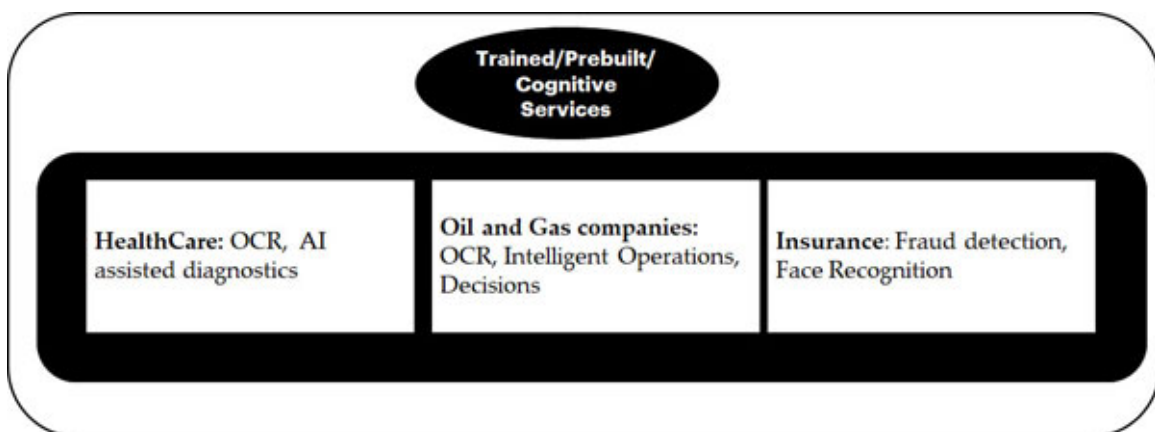
*Figure 1.3: AI services*

The following is an overview of AI Services:

## [Trained Services \(Prebuilt\)](#)

Trained Services are Microsoft Azure Cognitive Services that offer a set of machine learning-based Azure Rest APIs that can be easily integrated into applications to infuse intelligence.

- **Use cases for Trained/Prebuilt Services:** The following are a few use cases for leveraging Cognitive/Trained services:



*Figure 1.4: Trained/Prebuilt/Cognitive Services*

- **HealthCare:** We can leverage OCR and Read APIs and other custom algorithms/logic as appropriate to go through large volumes

of text that includes references to general entities (e.g. people's names) and domain-specific ones (e.g. drug and disease names) that need to be connected and related. Sometimes we also need to combine this with imagery that's analyzed in well-known ways (e.g. OCR) as well as apply leading-edge methods (e.g. AI-assisted diagnostics)

- **Oil and Gas companies:** Oil and Gas companies have teams of geologists and other specialists that need to understand seismic and geologic data. They often have a huge library of PDFs with pictures of samples over sample sheets full of handwritten field notes. They need to connect places, people (domain experts), events, and navigate all this information to make key decisions. We can leverage OCR and other custom algorithms/logic to analyse and make decisions.
- **Insurance:** Fraud Risk Analysis can be done by identifying anomalies in transactions using Azure Machine Learning, Face Recognition can be used as an additional authentication mechanism.

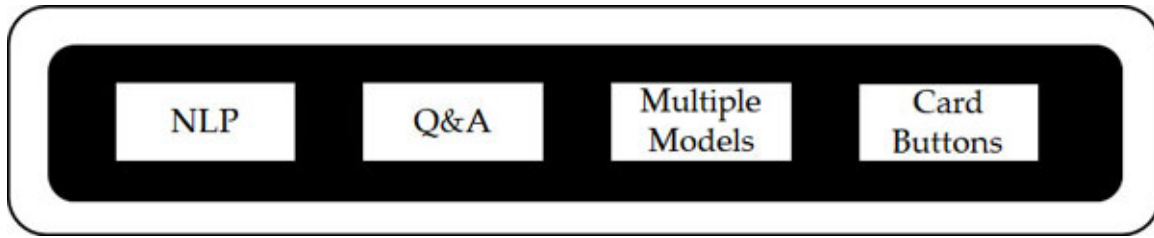
## Conversational AI

Conversational AI helps in creating AI-based conversational interfaces. Azure provides Bot service which includes Bot Builder SDK and tools for end-to-end bot development and Bot Connector service to connect to multiple channels. We can create bots in a number of languages.

A Bot is a web service that communicates using a conversational interface and leverages Bot Framework Service to exchange events and messages between the Bot and the channel.

Bot functionality can be extended by adding additional features like NLP etc. Microsoft Cognitive Services such as **Language Understanding Service (LUIS)**, can be added to the Bot interactions to make the conversation more intuitive.

The following are some ways of extending Bot functionality:



*Figure 1.5: Extending Bot Functionality*

The following table shows some of the additional features that can be leveraged to extend Bot functionality.

Feature	Description
Natural language processing	NLP can be used to recognize and understand the user's intent, catch spelling mistakes, enable speech etc.
Q & A	A knowledge base can be leveraged to make the conversation more natural.
Manage multiple models	'Manage multiple models' feature can be used to manage multiple models like LUIS, Q&A maker.
Add cards and buttons	Cards and buttons can be used to enhance user experience.

*Table 2.1: Leveraging Bot Functionality*

**Use cases for conversational AI:** A few use cases where conversational AI can be leveraged are as follows:

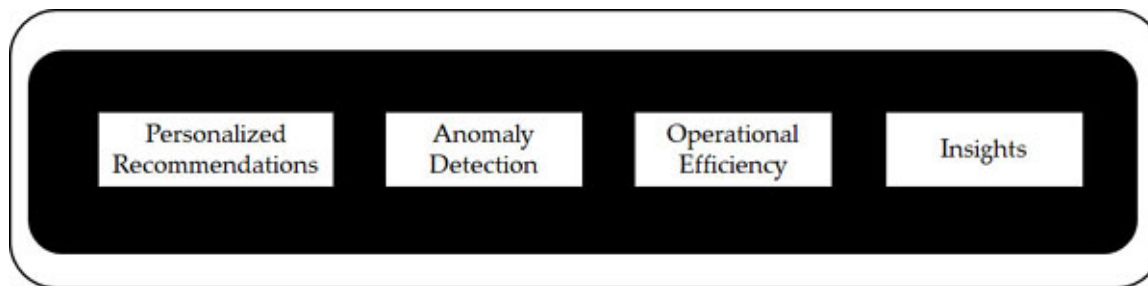
- Bots can help Insurers give their customers an easy way to look up the status of a claim and ask questions about benefits.
- Bots can help providers triage patient issues with a symptom checker, help patients find care, and look up nearby doctors.
- Bots can help end customers have intelligent conversations with banks /retail websites.
- Bots can be built leveraging language understanding models, bing knowledge and content out of the box. They are capable of learning from previous interactions.

## Custom services

Custom services help in building applications using custom-trained models. Product recommendation, fraud detection for transactions, sales prediction are some examples of applications that leverage custom services. Azure provides

an easy to use platform for developing/building, training and deploying custom machine learning models using tools like Visual Studio Code, Jupyter Notebooks and open source tools like Tensorflow, PyTorch, and so on. It is also easy to embed these as services in custom applications through simple REST API calls.

- **Use cases for custom services:** Given below are a few use cases where custom services can be leveraged



*Figure 1.6: Use cases for Custom Services*

- Provide product recommendations, personalized recommendations for medicines, apparel, products, and so on.
- Provide guidance and treatment to patients. Enable personalized suggestions and recommendations for checkups, and so on.
- Clinicians could proactively identify at-risk patients by analyzing their medical records, use data from smart devices to provide personalized care.
- Fraud identification using anomaly detection. Improved operational efficiencies. We can improve operational efficiencies by deriving intelligent insights and actions from data collected from devices.
- Improved safety. By providing insights into shop floor activities, the safety and efficacy of equipment can be improved by maintaining.

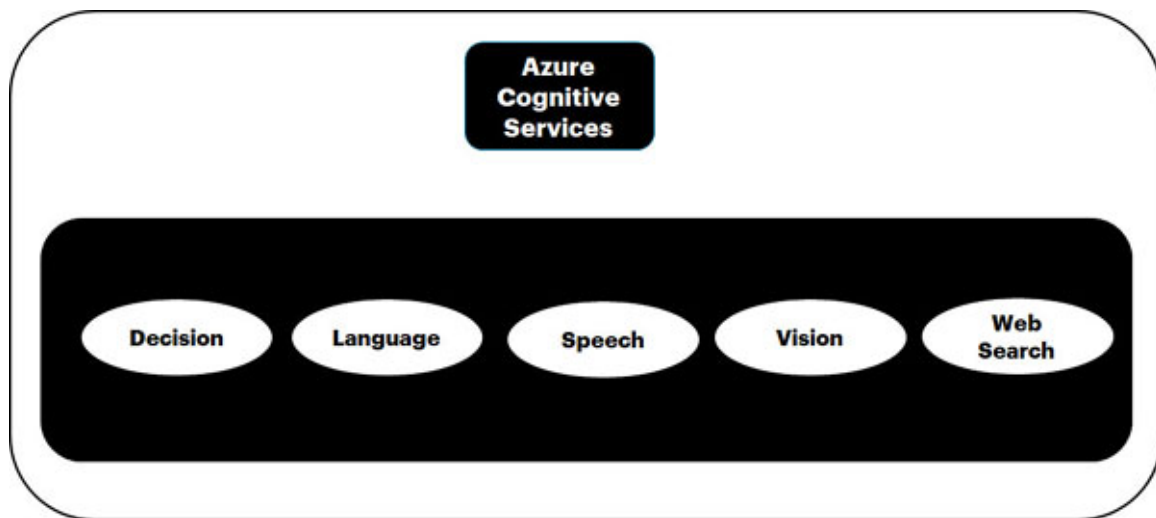
Now that we have an understanding of AI services we will go a little deeper into Azure Cognitive Services.

## [Azure Cognitive Services](#)

Azure Cognitive Services provide APIs, SDKs, and services that can be leveraged by developers to infuse cognitive intelligence like speech, optical character recognition, sentiment analysis, face recognition, video analytics, text analytics etc into applications.

Azure Cognitive Services offer a set of machine learning-based prebuilt Azure Rest APIs that are hosted in Azure. We can leverage these Azure Cognitive Service APIs / services to easily add intelligence into web, mobile, desktop applications such as face recognition, voice recognition, emotion, and sentiment detection, vision and speech recognition, knowledge, search, and language understanding. These APIs are cross-platform and are accessible across devices.

Azure Cognitive Services can be broadly classified into Decision, Language, Speech, Vision and Web Search as depicted here:



*Figure 1.7: Azure Cognitive Services*

Here is a brief description of different services available under Azure Cognitive Services:

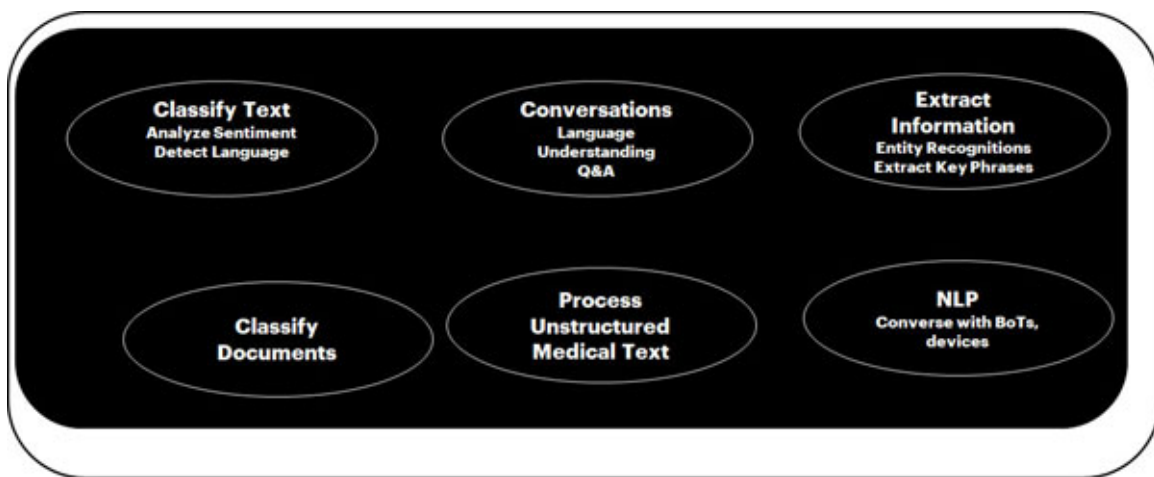
- **Decision Service:** Helps build apps that help add recommendations for efficient decision-making. Decision Services include Personalizer and Content Moderator.
  - **Personalizer service:** Personalizer service is a cloud-based service that helps applications choose the best content to surface to users and helps in reinforcement learning through easy-to-use APIs. Some of the use cases for decision API include Intent clarification and disambiguation which help users have a better experience when their intent is not clear by providing an option that is personalized. Decision API helps provide suggestions for personalized menus and options.



For example, it can be used to suggest a product to shoppers. The application can monitor the users' reaction and send back a reward score to the personalizer service thus enabling continuous feedback and improvement of the machine learning model, which further enhances the personalizer's ability to contextualize recommendations.

- **Anomaly detector:** Anomaly detector helps in monitoring and detecting anomalies. Anomaly detector takes time-series data as input and helps in detecting dips, spikes in the data using anomaly detection algorithms. It can be used in the cloud or edge and can be customized.
- **Content moderator:** Content moderator helps in monitoring and detecting offensive content.
- **Language service:** Helps build apps that embed conversational language understanding. This helps in leveraging intelligence to decipher a users' conversation to retrieve information.

The following are some of the use cases for leveraging language service.



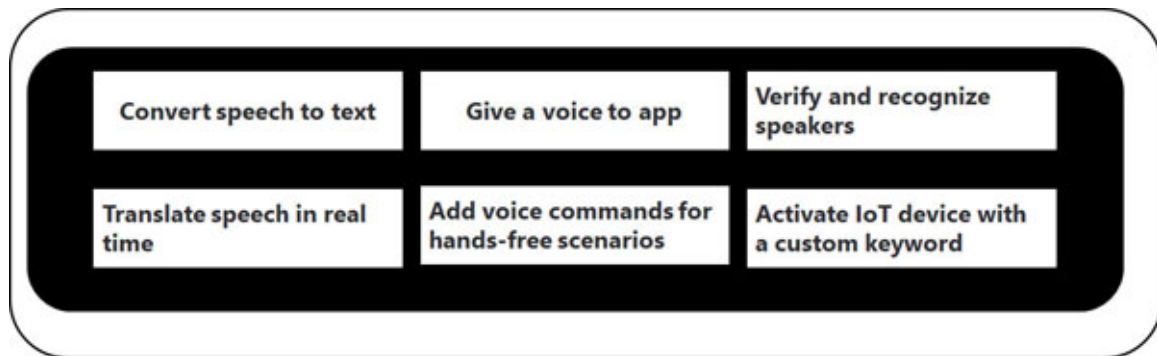
*Figure 1.8: Language Service*

It helps in identifying entities and their relationships, better understand sentiment and comprehend information much faster. It enables usage of domain specific labels and enables conversations with Bots, applications, IoT Devices using Natural Language Processing.

- **Speech service:** Helps convert speech into text and text into natural-sounding speech. Speech helps in translating from one language to

another and enables speaker verification and recognition.

Given below are some of the use cases for leveraging Speech Service:



*Figure 1.9: Speech Service*

- **Speech to Text:** Helps translate audio streams to text in real time
- **Text To Speech:** Helps convert text into synthesized speech
- **Speech Translation:** Helps enable real time, multi-language translation of speech
- **Voice Assistants:** Helps empower developers to create conversational interfaces to applications
- **Speaker Recognition:** Helps provide APIs to recognize and identify speakers
- **Intent Recognition:** Helps recognize speech, entities and intent
- **Vision Service:** Helps in analyzing an image or a video and perform certain analysis and derive insight such as recognizing, identifying caption, abusive content, and so on.
  - **Computer Vision:** Provides access to advanced algorithms for processing images and returning information.
  - **Face:** The Face service provides access to advanced face algorithms, enabling face attribute detection and recognition.
  - **Custom Vision Service:** Provides capabilities to build image classifiers.
- **Web Search:** Helps add Bing Search APIs to apps. Bing Search helps in searching information from webpages, images, videos, and news with a single API call. Helps provide instant answers to user queries which are easily configured to include web pages, images, videos, news, translations as JSON based on search relevance and your Bing Web

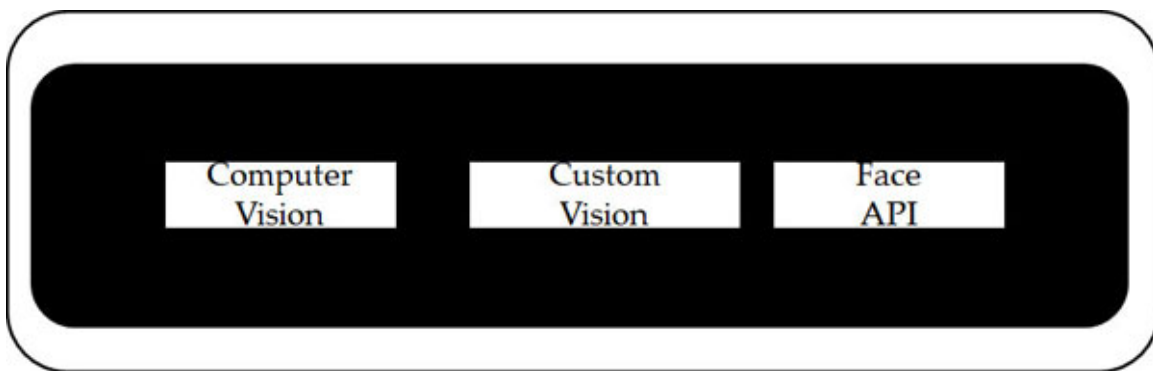
Search subscriptions. It helps in suggesting search terms in real time, filter results, localize search results and analyse search metrics with Bing Statistics.

In this book, we will focus on Azure Cognitive Services as described above and Azure Applied AI Services. We will start with understanding the basics and then get into details on each of the APIs provided by Azure Vision API.

## Azure Vision Service

Azure Vision Service provides APIs / SDK to analyze images, videos, extract text from images, and moderate content in images. We can either leverage the APIs or the SDK. Vision APIs help infuse into applications vision capabilities like image processing, video analytics, and insights, face recognition and detection capabilities, OCR capabilities into applications.

Here is a brief description of each of the APIs provided by Azure Vision Service:



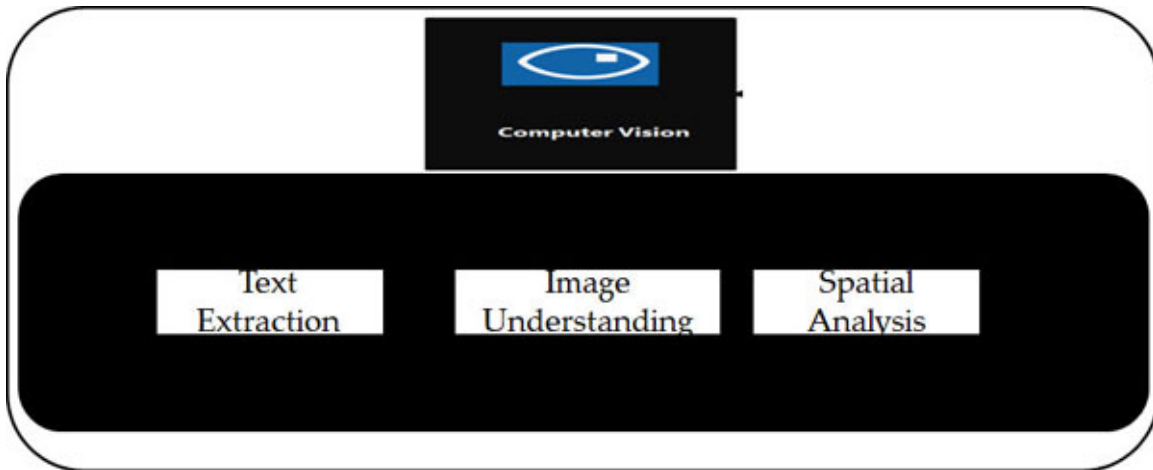
*Figure 1.10: Azure Vision Service*

We will now look at each of the APIs in detail.

- **Computer Vision**

Computer vision provides APIs for analyzing images, extracting text from images and moderating content in images. Azure Computer Vision provides APIs to help in processing and analyzing photos, images, and videos. These APIs allow the users to perform the classification of images, extract sentiment from images and read handwriting and convert to text or add descriptions to images based on the intent in the images. It also helps in recognizing human faces and applying tags based on classification, and so on. For example, Computer Vision can help

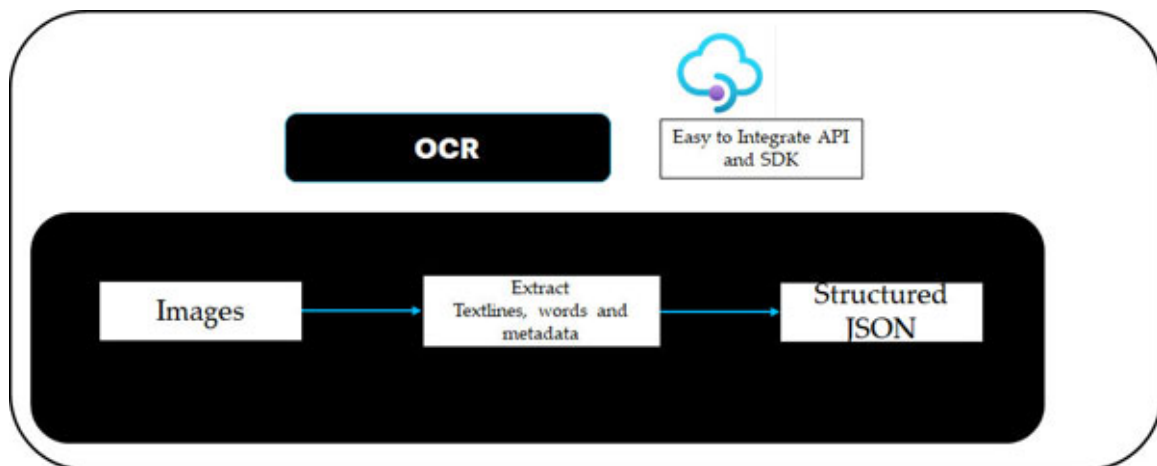
determine if an image contains adult content, can find all of the human faces in an image, and can draw insights from a video.



*Figure 1.11: Azure Computer Vision*

A detailed description of the APIs is given as follows:

- **Text Extraction:** Helps in extracting text from Images. It provides Optical Character Recognition which includes Read API and OCR API.



*Figure 1.12: Use cases for Custom Services*

- Read API helps in extracting text from images and is optimized for text-heavy images and documents which have a lot of text content. We can use this API to extract information from business cards, posters, receipts, whiteboards, and so on.
- OCR API executes synchronously and is not optimized for large documents. It leverages an older recognition model. OCR supports around 25 languages. OCR automatically detects the language of the

detected text. It also provides the ability to correct the rotation of the recognized text and provides the frame coordinates of each word.

- Analyze Images:

Tag Visual Features	Detect /Brands	Detect Brands
Categorize Images	Describe Image	Detect Faces
Detect Image Types	Detect domain-specific content	Detect the color scheme
Generate a thumbnail	Get the area of interest	Content Moderator

*Figure 1.13: Use cases for Custom Services*

- Image Recognition and Analysis can be leveraged for extracting various visual features from images like categorization, image type. Image Analysis can be performed by leveraging the Analyze Image API which provides features like locating all the faces in an image, checking if an image contains adult content, describing an image, generating images thumbnails, identifying and tagging visual features in an image, detecting objects in an image, detecting commercial brands in an image, identifying and tagging an entire image, detecting a color scheme, detecting domain-specific content, and getting the bounding box coordinates of interest.
- Here is a brief overview of the feature of Analyze Images API:

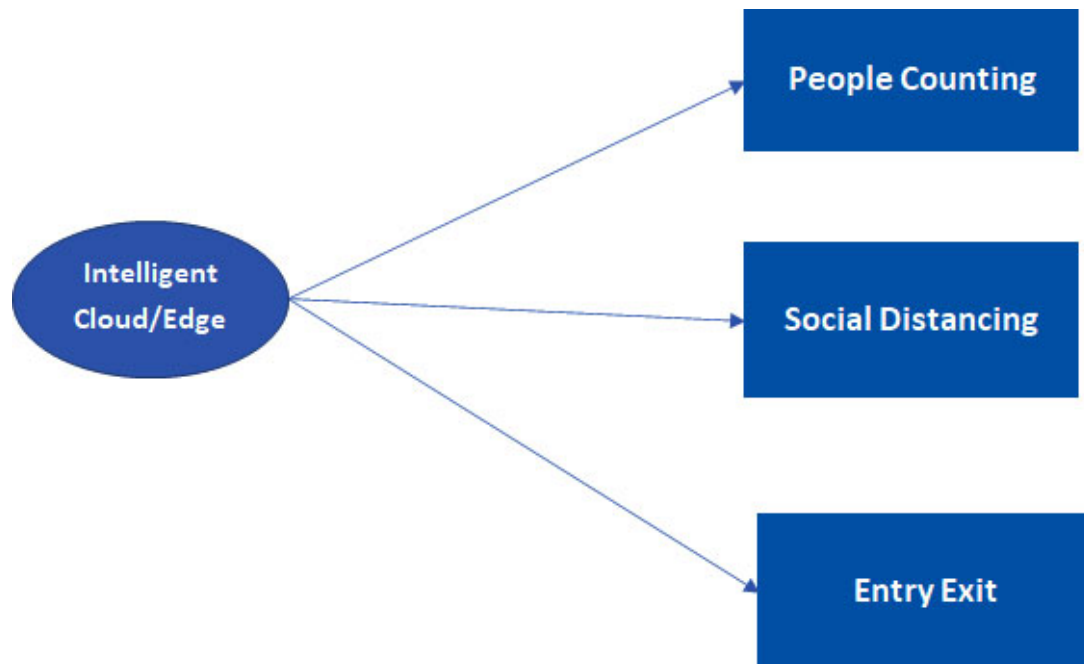
Action	Description
Tag visual features	Helps in identifying and tagging visual features in an image leveraging a large number of pre-built models.
Detect objects	Helps in returning the coordinates of the bounding boxes for each tag. We can leverage this to process relationships between objects in a given image.

Detect brands	Helps in detecting commercial brands in an image. This helps in checking the popularity of brands.
Categorize an image	Helps in identifying and categorizing images using pre-built categories.
Describe an image	Helping in returning descriptions based on the objects identified in an image. There is a confidence score also returned for each object identified.
Detect faces	Helps in detecting faces and providing information about each face detected. We can use this information for the implementation of digital id or pose detection etc.
Detect image types	Helps in identifying if the given image is a line or a clip art.
Detect domain-specific content	Helps in identifying domain-specific content using pre-built models like celebrities.
Detect the color scheme	Helps in identifying whether the image is black and white or color and helps in identifying the accent and dominant colors for a color image.
Generate a thumbnail	Helps in identifying and cropping areas of interest and returning a thumbnail. It can identify faces etc in an image and crops the same. It provides the best quality thumbnail. It has the ability to identify the face etc in an image, and crop around it.
Get the area of interest	Helps in analyzing the content and returning the coordinates of the area of interest.
Content Moderator	It helps detect adult and offensive content in an image and returns confidence scores. The threshold for flagging content can be set as per the requirement. Content Moderator APIs help in content moderation for videos, text, and images. The Content Moderator APIs help with labeling content that is offensive. This flagged content can be handled by workflow-based applications to either approve or delete the offensive content which might also require human intervention.

**Table 2.2:** *Analyze Images*

- **Spatial Analysis:** Computer Vision Spatial Analysis helps in extracting insights and generating events by ingesting streaming video from cameras. It detects people's presence and their movements and helps in counting the number of people entering a space or measuring compliance with face masks etc. It can also help in space utilization optimization solutions.

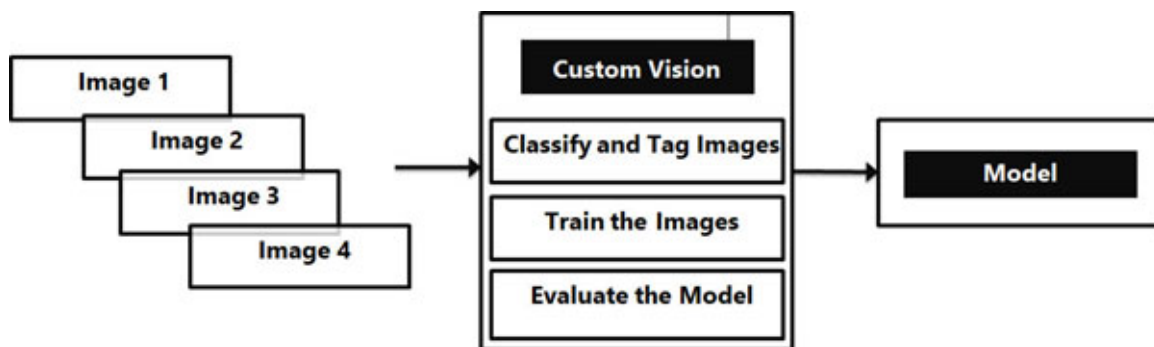
Following are some use cases for Spatial Analysis:



*Figure 1.14: Use cases for Spatial Analysis*

- **Custom Vision Service:** Custom Vision provides custom training of images for use cases like anomaly detection, objects with custom tags defined detection etc. With Custom Vision we can build and train our own custom vision model. The more the images are fed into the model and the model is trained, the more accurate the prediction becomes as the model learns to discern the images.

The following is how custom vision models are trained:



*Figure 1.15: Custom vision model*

Once a project is created, we can upload images, define tags and then train the images against the tags. Good accuracy is obtained by uploading at least 50 images in each category. The model needs to be retrained when more images are uploaded.



Custom Vision Service provides two APIs to integrate the custom modes into applications:

Prediction API and Training API. These are defined as described below:

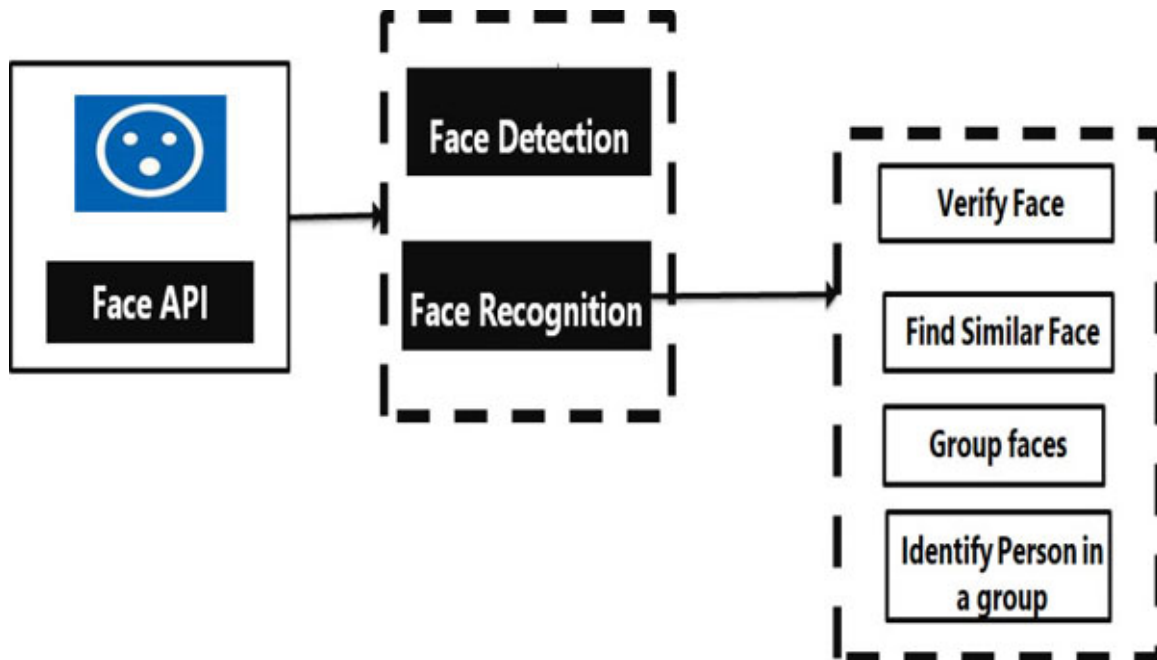
- **Prediction API:** Prediction API allows us to integrate the trained model into our custom applications. This API can be accessed through REST services or by using a client library in the case of Windows applications.
- **Training API:** Training API allows us to build and train images using code and can be integrated into applications.
- **Pricing:** Visit the following link for more details on pricing:

<https://www.azure.microsoft.com/en-gb/pricing/details/cognitive-services/customvision-service/>

While the free tier is used for experimentation, the pricing is based on storage and other factors like transactions per second etc.

- **Face API:** Face API makes it possible to detect, analyze, identify, organize and tag faces in pictures. It also allows identifying a mood or emotion with around 9 or 10 emotions currently being supported.

The Azure Face API helps in analyzing faces, identifying people, and finding similar faces. Face Detection and Face Recognition are two basic aspects covered in Face API. It also helps in detecting emotions like anger, fear, surprise, disgust, happiness, and sadness in an image.



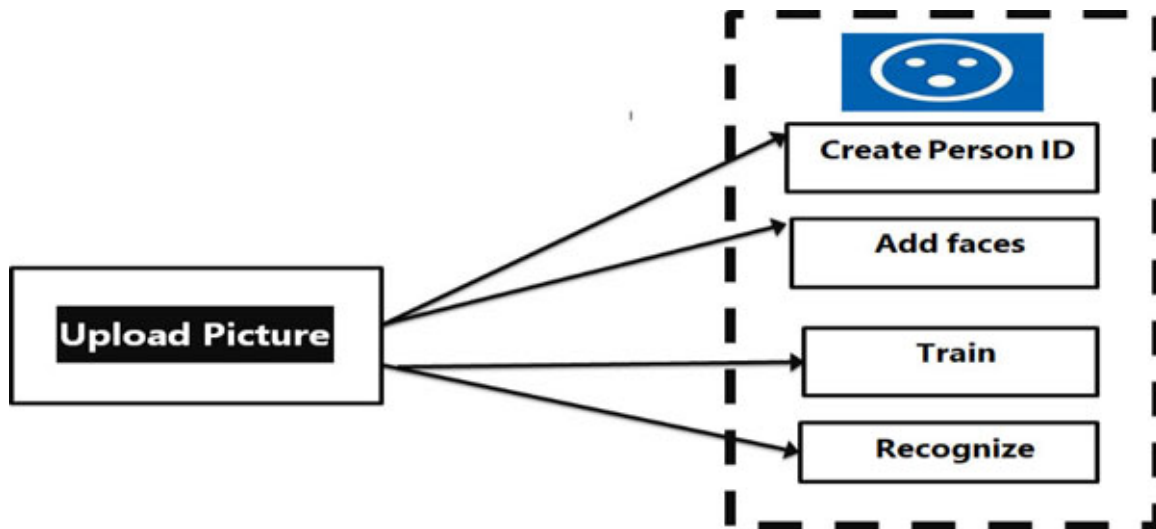
*Figure 1.16: Face API*

Face Detection helps discover attributes of a face such as gender, age, and so on.

Face Recognition supports the following subcategories of functionalities:

- **Face Verification:** This verifies if two faces match Finding Similar Face: This takes a face and tries to create an order based on the degree of similitude in a set of faces.
- **Face Grouping:** This takes a set of photos and creates subgroups of faces based on their degree of similarity.
- **Face Identification:** This allows us to train images to be part of a group. We can train the model with more and more images and improve the accuracy. We have to create a PersonGroup object which will have multiple images of that person and train.

The following image shows how we can train faces for face recognition:



*Figure 1.17: Face Recognition*

## [Accessing the APIs](#)

A subscription key is required to invoke Computer Vision API which can be passed through a query string parameter specified in the request header. We can obtain the subscription keys as follows:

Given below is the link for Computer

Vision: <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>

Once we have reached this screen, click on 'Try the Computer Vision API'. We then need to select the subscription type.

Once we get the subscription keys we can leverage them to invoke the API from custom code.



*Figure 1.18: Subscription keys*

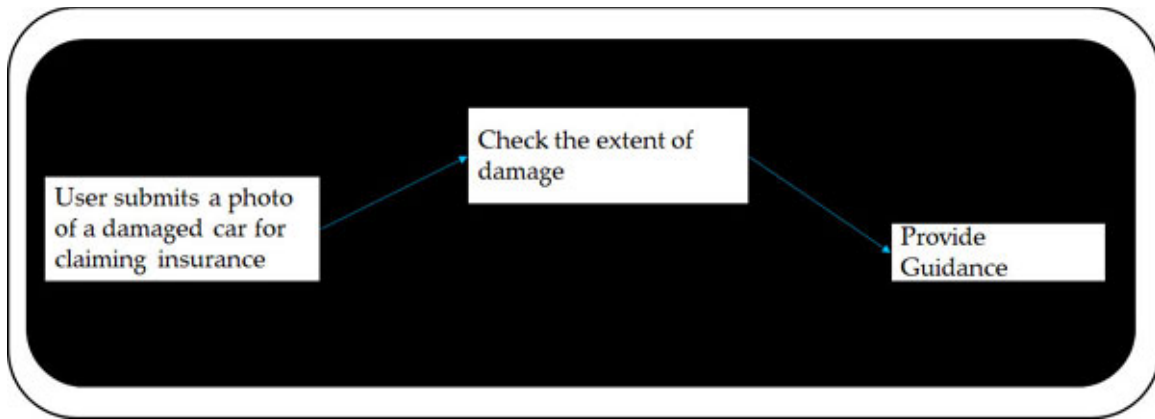
We will explore some of the use cases that use Azure Vision APIs.

## Use cases

Azure Vision APIs can be used in a myriad of enterprise scenarios that leverage Image Classification, Object Detection, Emotion / Sentiment Analysis. Sentiment analysis can be used to analyze the sentiment.

Here are two examples:

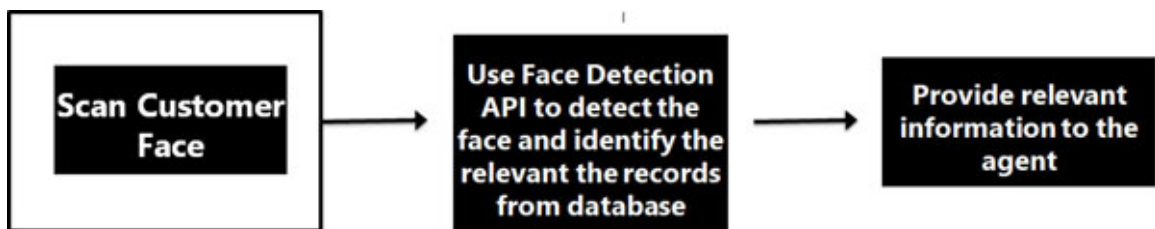
- **Smart Image Classification and Object Detection:** Images can be classified as good or bad based on image classification. This can be useful in use cases such as traffic jams, sending traffic alerts, monitoring homes, and safety solutions.
- Insurance companies can use smart image classification to classify images and guide insurance agents on the right premium amount.



*Figure 1.19: Use case for Insurance*

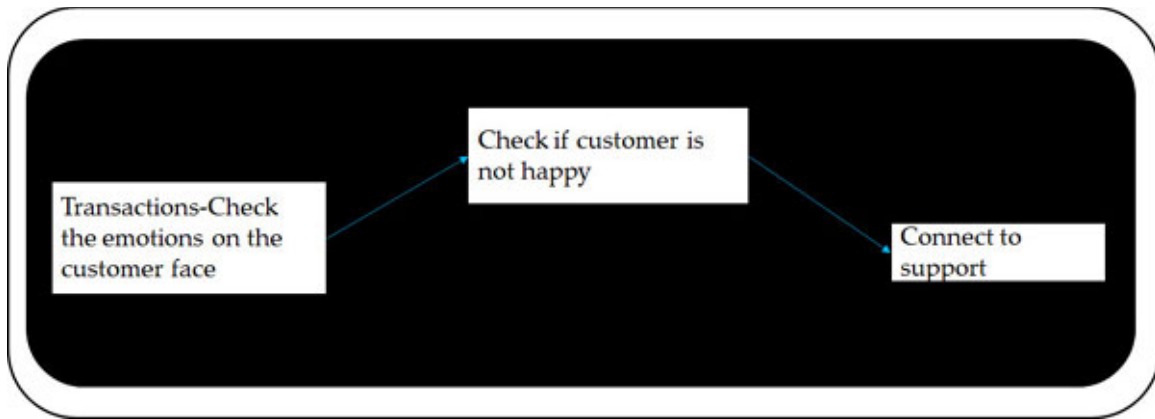
- **Smart Security:** A Face Detection API can be leveraged to identify the customer and pull up relevant information such as electronic health records for providing the right service. Example scenarios could include a patient waiting at a hospital counter or a customer waiting at a bank counter.

Given below is a use case in the hospital/bank counter. When a customer comes to the counter, the face can be scanned and the relevant record pulled up so that the nurse or teller knows the history of the patient before his / her turn comes. This will help in reducing the wait time.



*Figure 1.20: Use case in the hospital counter*

- **Emotion Analysis** can help organizations to understand how their product/services are received by their customers and help them improve their services. It can also help with crowd analytics. If many people are looking bored in a meeting, maybe it is time to announce a break.



*Figure 1.21: Use case for Emotional Analysis*

- **Content Moderator** can help automate tasks to act on inappropriate content so that content sharing sites are safe and easy to use. Video Indexer can then help with home monitoring and the translation of movies or videos into other languages (this is beneficial in the case of knowledge sharing in multiple languages).

## Setting up Developer Tools

We can leverage Visual Studio Code or Visual Studio for the lab exercises discussed in the chapters.

VS Code is a free code editor which is cross-platform. It runs on Windows, macOS and Linux, operating systems.

Download Visual Studio Code from

### **Download Visual Studio Code - Mac, Linux, Windows**

The code for the examples is placed in the following GitHub location:

**Email/UserName:** [azurevisionbook@gmail.com](mailto:azurevisionbook@gmail.com)

**Password:** azurevisionbook@123

## Conclusion

In this chapter, we have looked at the basics of Cognitive Services as well as services and APIs available under Vision Services. We have also learned how to use these APIs in enterprise scenarios.

In the next chapter, we will be exploring building custom applications using Vision API. This includes developing custom applications leveraging Image Processing, Face Recognition.

## **Multiple choice questions**

This section consists of a set of questions that are designed to test your knowledge of the concepts covered in this chapter.

When answering these questions, imagine there is an audience who has watched a movie. You have images of the movie and images of the audience coming out of the theatre.

- 1. You want to group people in the images. Which API will you use?**
  - a. Face Detection
  - b. Face Recognition
  - c. Face Grouping
  - d. Face Verification
- 2. You have been asked to analyze the emotions of the people in the image. Which API will you use?**
  - a. Emotion
  - b. Face
  - c. Analyze
  - d. Verification
- 3. You have been asked to look at the image of the movie title and recognize the text in the image. Which API will you use?**
  - a. Text Recognition
  - b. OCR
  - c. Read API
  - d. Image Detection
- 4. Which API will you use to create a custom model to train images?**
  - a. Computer Vision
  - b. Custom Vision
  - c. Object Detection
  - d. Image Classification
- 5. You have been asked to look at an image and analyze its attributes. Which feature or API will you use?**

- a. Computer Vision
  - b. Custom Vision
  - c. Object Detection
  - d. Image Recognition
- 6. Which APIs can help automate tasks to act on inappropriate content so that content sharing sites are safe and easy to use.**
- a. Sharing
  - b. Content Moderator
  - c. Ink Recognizer
  - d. Digital Recognizer
- 7. You have been asked to analyze the emotions in the video. Which API will you use?**
- a. Face Detection
  - b. Video Indexer Insights
  - c. Face Grouping
  - d. Face Verification
- 8. Which API will be used to read from a receipt?**
- a. Text Recognition
  - b. Form Recognizer
  - c. Receipt Recognizer
  - d. Image Recognizer
- 9. Which API will help build relations between fields and entries in documents?**
- a. Text Recognition
  - b. Form Recognizer
  - c. Receipt Recognizer
  - d. Image Recognizer
- 10. You have been asked to detect commercial brands. Which feature or API will you use?**
- a. Computer Vision

- b. Custom Vision
- c. Object Detection
- d. Image Recognition

## **Answers**

You have images of the movie and images of the audience coming out of the theatre.

1. **c.**
2. **b.**
3. **b.**
4. **b.**
5. **a.**
6. **b.**
7. **b.**
8. **b.**
9. **b.**
10. **a.**



# CHAPTER 2

## Azure Computer Vision - Image Analysis, Processing, Object Detection

In this chapter, the reader will understand how to use Azure Vision APIs for developing custom applications that can be used for processing images and detecting objects.

### Structure

This chapter will cover the following topics:

- Image processing:
  - Image analytics, content moderation using Azure Computer Vision API
  - Image classification using Azure Custom Vision Service
  - Object detection using Azure Custom Vision Service

### Objective

In this chapter, the reader will understand how to use Azure Vision APIs for developing custom applications that can be used for processing images and detecting objects. After reading this chapter, the reader will have the skills that make it possible to effectively build applications by leveraging image analytics, content moderation, and Azure Custom Vision Services.

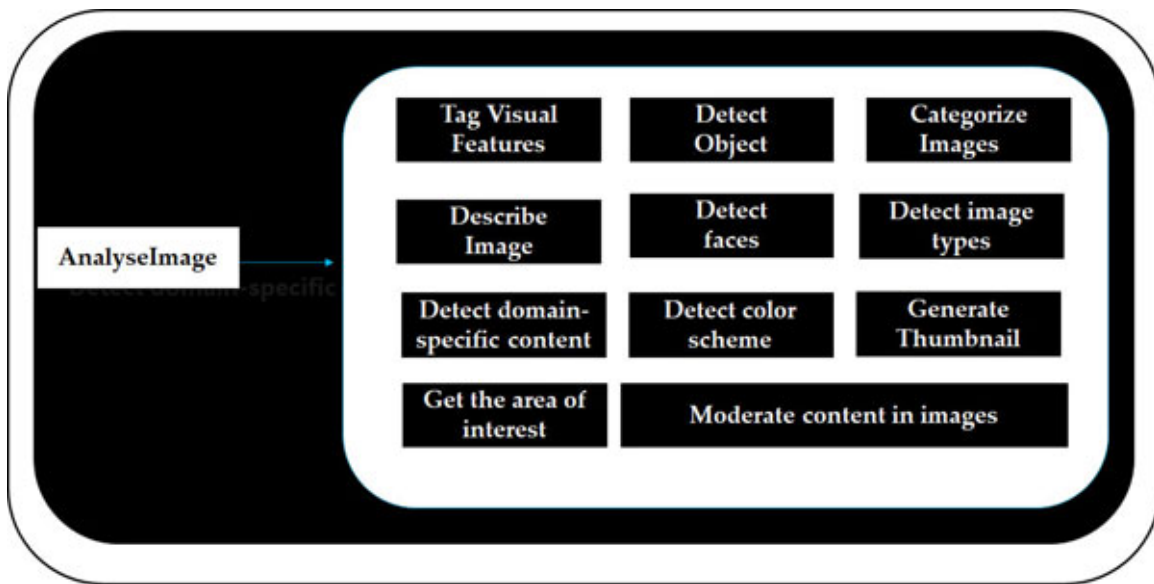
### Image processing

Processing image information is a very important aspect of applying artificial intelligence while developing business-critical applications. We can analyze and process images using image analytics and image classification which are two different aspects of Azure Vision Service. The following sections cover image analytics and image classification in detail.

## Image analytics - Azure Computer Vision API

We can leverage Azure Computer Vision Analyze Image API to analyze an image and obtain information on the visual aspects of an image. It returns descriptions based on different visual features, with a confidence score against each feature. The final output is a list of descriptions ordered from highest to lowest in confidence. Based on thousands of recognizable objects, computer vision returns tags based on the objects, living beings, and actions identified in the image in addition to details on the background. Hints are provided wherever the information is not available.

The following image shows the actions we can perform using Analyze Image API:



*Figure 2.1: Actions we can perform using AnalyzeImage API*

### **Analyse API in detail:**

- **Detect object:** We can leverage object detection API to list objects and the corresponding coordinates of the objects and to check the number of similar instances of an object. We can leverage detect operation to figure out the inter-object relationships and also to get the coordinates for the bounding box for each object found. The tags are applied based on objects and living things included in the image.

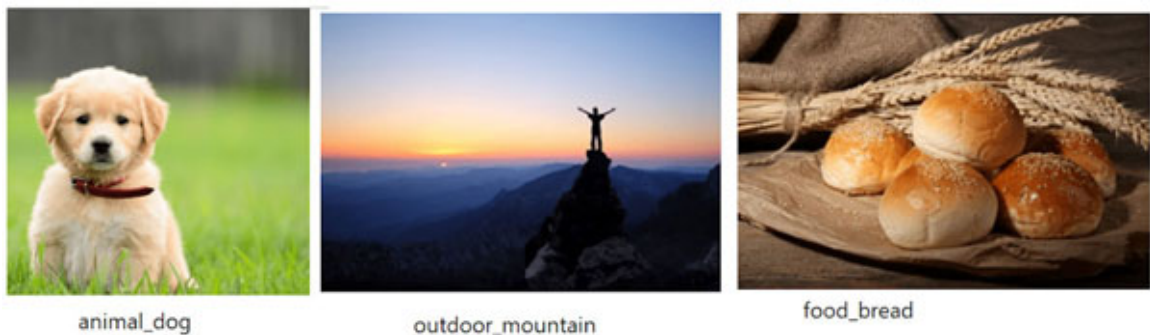
For instance, for an image containing objects like cat, dog etc., Detect API will list those objects together with their coordinates in the image. This API can be used to correlate the relationships between the objects in

an image. Detect API also helps in checking if multiple instances of the same tag exist in an image.

Here are some limitations of Detect API

- Objects which are less than 5% of the image are not detected
- Objects which are organized too close to each other might not be detected
- **Categories:** We can leverage this method to get categories from about 86 categories.

Example categorizations include dog, outdoor, bread etc.:



*Figure 2.2: Example categorizations*

- **Detect Brands:** We can leverage this method to derive insights on which brands are most popular on social media.



*Figure 2.3: Example Brands*

**Categorize an image:** We can leverage this method to identify and categorize an entire image, using a category taxonomy with parent/child hereditary hierarchies.

- **List domain-specific models:** We can leverage this method to return the domain-specific models supported by Computer Vision API which include celebrity recognizer, landmark recognizer.
- **Generate a thumbnail:** We can leverage this method to return a thumbnail image with the width and height specified by the user. The image is analyzed by the service. **Region of Interest (ROI)** is identified and smart cropping coordinates are generated based on the ROI.
- **Describe image:** We can leverage this method to return a description of an image which is a collection of tags for the content in the image with a confidence score. Input methods supported include specifying an image or an image URL. We have a lab in the subsequent section.
- **Detect faces:** We can leverage this method to provide information about faces detected in an image. Details for each face returned include age, coordinates, gender, and rectangle.
- **Detect image types:** We can leverage this method to detect characteristics about an image, such as whether an image is a line drawing or the likelihood of whether an image is clip art.
- **Detect the color scheme:** We can leverage this method to analyze an image and determine if an image is black and white or color. For color images, this method also identifies the dominant and accent colors.
- **Get the area of interest:** We can leverage this method to analyze the contents of an image and return the coordinates of the area of interest.

We will now go through a sample lab where a sample image is analyzed using AnlaysiaImage API.

## [Image Analytics -Lab](#)

We will create a sample project using Azure custom vision service. We will analyze the image using Azure Computer Vision API.

Here are the prerequisites and steps required to run the example:

### **Prerequisites**

- A valid Computer Vision subscription key is required.
- We can use a free trial subscription key which can be obtained using Try Cognitive Services or we can subscribe to Computer Vision and get a key

using Create a Cognitive Services account.

- Visual Studio 2017 and above
- NuGet package to be added: Microsoft.Azure.CognitiveServices.Vision.ComputerVision client library

Follow the steps given below to work on the Image Analytics example:

### **Step 1: Create and run the sample application: Image Analysis using Computer Vision API**

We have to log into the Azure Portal. We have to click on ‘**Create a Resource**’. We have to select ‘**AI + Machine Learning**’. We have to select ‘**Computer Vision**’.

We have to provide the following details:

- **Name:** We have to provide the name for the Computer Vision API.
- **Subscription:** We have to select an active subscription
- **Location:** We have to select the location of the resource group.
- **Pricing Tier:** We have to select the pricing tier.
- **Resource Group:** We have to create or select an existing resource group.

**Step 2: Endpoint:** We can choose available locations while creating a Computer Vision API. We can choose the nearest region. We have chosen Central India for this tutorial.

The image shows the Azure portal interface for creating a new Computer Vision resource. On the left is a dark sidebar with navigation links: 'Create a resource', 'Home', 'Dashboard', 'All services', 'FAVORITES', 'All resources', 'Resource groups', 'App Services', 'Function Apps', 'SQL databases', 'Azure Cosmos DB', 'Virtual machines', and 'Load balancers'. The main area is titled 'Create' with a sub-header 'Computer Vision'. It contains several configuration fields, each with a red asterisk indicating it is required:

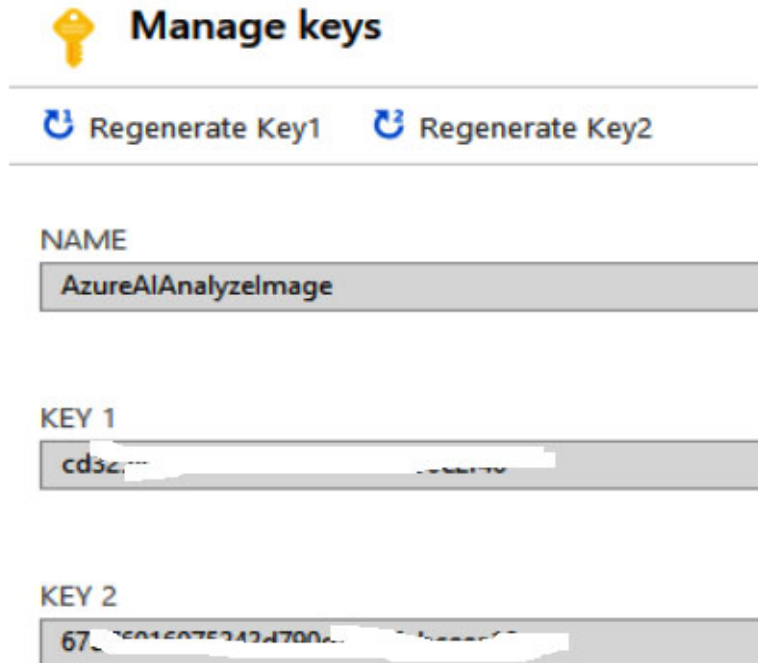
- Name:** A text input field containing 'AzureAIAnalyzelImage' with a green checkmark on the right.
- Subscription:** A dropdown menu showing 'Sponsorship\_Cloud\_Experience'.
- Location:** A dropdown menu showing 'Central India'.
- Pricing tier:** A dropdown menu showing 'F0 (20 Calls per minute, 5K Calls per month)' with a link '(View full pricing details)' in blue.
- Resource group:** A dropdown menu showing '(New) AzureAI'.

Below the resource group dropdown is a blue link that says 'Create new'.

*Figure 2.4: Endpoint Configuration*

**Step 3:** Obtain subscription keys:

Get trial keys by going to **Manage Keys**:

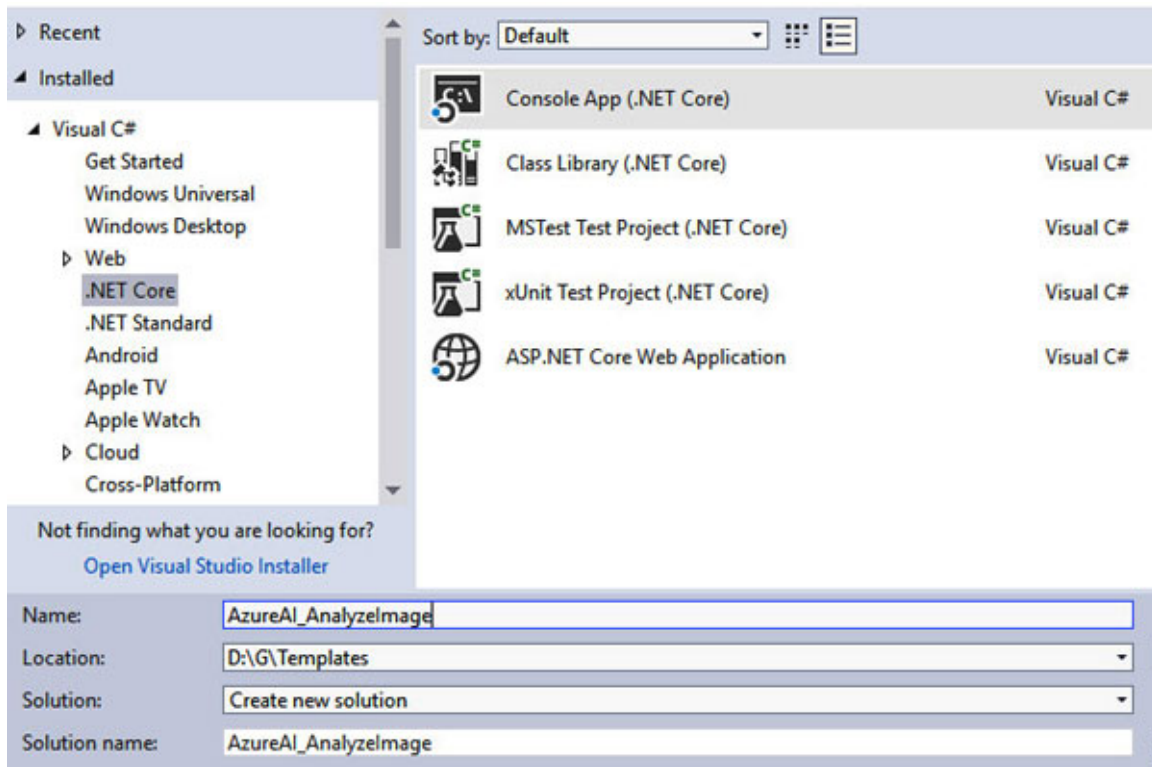


*Figure 2.5: Getting Trial Keys*

**Step 4: Create a sample console application to analyze an image:**

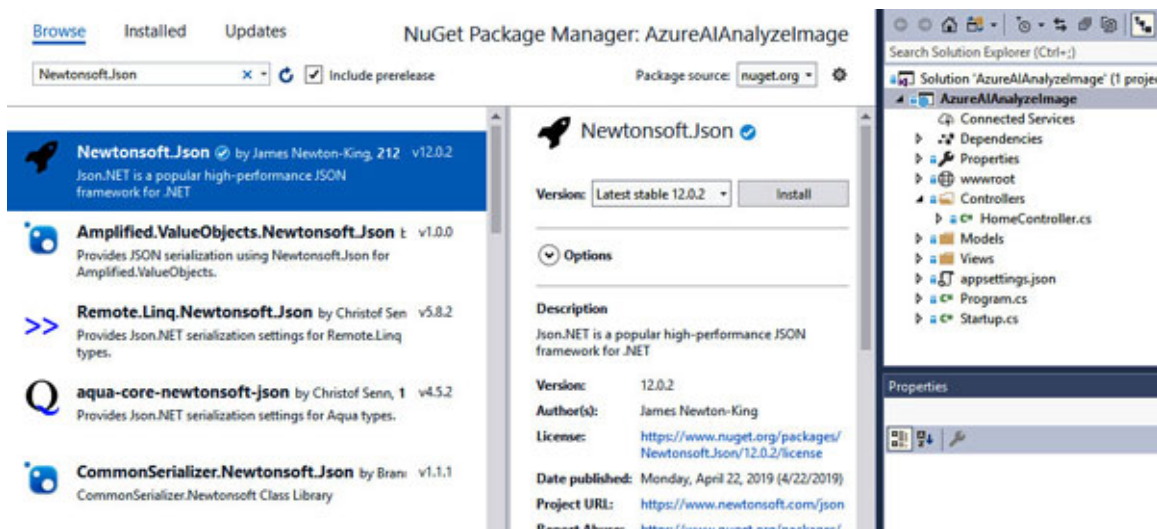
Create a new Visual Studio solution in Visual Studio, using the Visual C# .NET Core Console App.





*Figure 2.6: APIs provided by AnalyzeImage API*

**Step 5:** Install Newtonsoft.Json NuGet package.



*Figure 2.7: Installing Nuget Package*

**Step 6:** Replace the code in Program.cs with the following code:

```
using System;
using System.IO;
```



```
using System.NET.Http;
using Newtonsoft.Json.Linq;
Using System.Net.Http.Headers.
Using System.Threading.Tasks;
```

Replace the value of **computerVisionSubscriptionKey** with your subscription key. Replace the value of **baseURL** with the endpoint URL for the **Analyze Image** method from the Azure region where you obtained your subscription keys.

```
namespace AzureAI.ImageProcessing_AnalyzeImage
{
    internal class Program
    {
        // Replace computerVisionsubscriptionKey with your
        subscription key
        private const string computerVisionsubscriptionKey =
        "Your SubscriptionKey";
        // Replace computerVisionsubscriptionKey with the
        // correct Azure region.
        private const string baseURL =
        "https://centralindia.api.cognitive.
        microsoft.com//vision/v2.0/analyze";
        private static void Main(string[] args)
        {
            //Prompt the user for an input image file
            Console.Write("Enter the path of the image: ");
            string filePath = Console.ReadLine();
            if (File.Exists(filePath))
            {
                Console.WriteLine("\n Results are getting processed....\n");
                ProcessImage(filePath).Wait();
            }
            else
            {
                Console.WriteLine("The given file path is invalid");
            }
            Console.WriteLine("\nPress Enter to exit the program...");
            Console.ReadLine();
        }
    }
}
```

```

/// <summary>
/// The below function analyses the image
/// <param name="filePath">The name of file to be
analyzed</param>
///
private static async Task ProcessImage(string filePath)
{
    try
    {
        HttpClient client = new HttpClient();
client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-
    Key", computerVisionssubscriptionKey);
        string requestParameters = "visualFeatures=Categories,
        Description,Color";
        string uri = baseUrl + "?" + requestParameters;
        HttpResponseMessage response;
        byte[] byteData = GetPictureAsByteArray(filePath);
        using (ByteArrayContent content =
            new ByteArrayContent(byteData))
        {
            content.Headers.ContentType = new
                MediaTypeHeaderValue("application/octet-stream");
            response = await client.PostAsync(uri, content);
        }
        string jsonResponse = await
            response.Content.ReadAsStringAsync();
        Console.WriteLine("\nResponse:\n\n{0}\n",
            JToken.Parse(jsonResponse).ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("\n" + e.Message);
    }
}

private static byte[] GetPictureAsByteArray(string
filePath)
{
    using (FileStream fileStreamReader = new
        FileStream(filePath, FileMode.Open, FileAccess.Read))

```

```

{
    // Read the file's contents into a byte array.
    BinaryReader binReader= new BinaryReader(fileStreamReader);
    return binReader.ReadBytes((int) fileStreamReader.Length);
}
}
}
}

```

The sample input file is provided as follows:



*Figure 2.8: Sample Input for Image Processing*

**Output:** The response captures the image to contain dogs with a confidence level of 0.9921875.

The response returns a caption of “a dog sitting in the grass” with a confidence of 0.737724174239289

Tags captured include “dog”, “grass”, “sitting”, “animal”, “brown”, “laying”, “front”, “yellow”, “large”, “standing”, “white”

Full response captured is as follows:

```

{
  "categories": [
    {

```

```
    "name": "animal_dog",
    "score": 0.9921875
  }
],
"color": {
  "dominantColorForeground": "White",
  "dominantColorBackground": "White",
  "dominantColors": [
    "White",
    "Green"
  ],
  "accentColor": "BB8310",
  "isBwImg": false,
  "isBWImg": false
},
"description": {
  "tags": [
    "dog",
    "grass",
    "sitting",
    "animal",
    "brown",
    "laying",
    "front",
    "yellow",
    "large",
    "standing",
    "white"
  ],
  "captions": [
    {
      "text": "a dog sitting in the grass",
      "confidence": 0.737724174239289
    }
  ]
},
"requestId": "550b3576-f360-42b3-aea8-70799c685852",
"metadata": {
  "width": 781,
```

```
    "height": 512,  
    "format": "Png"  
  }  
}
```

Now we will go through the Content Moderator API which can be used to moderate content.

## **Image Classification using Azure Custom Vision Service**

We can leverage Microsoft Custom Vision service to build our own image classification model to classify images into categories as per tags. With almost zero coding we can upload images, train, test and publish the models. Once the models are tested they can be invoked in custom applications to classify an image using prediction key and prediction URL.

## **Azure Custom Vision Service-Image Classification Lab**

The following is a lab describing a custom vision where we will use images of bags to categorize and train and publish models.

We will create a sample project using Azure custom vision service. We will upload and train images. We will train the model and then use it for prediction.

### **Step 1: Prerequisites**

- Go to <https://www.customvision.ai/>
- Create a New Project – Select **Classification**

The screenshot shows the 'Create new project' dialog box. It has a title bar with a close button (X). The form contains the following fields and options:

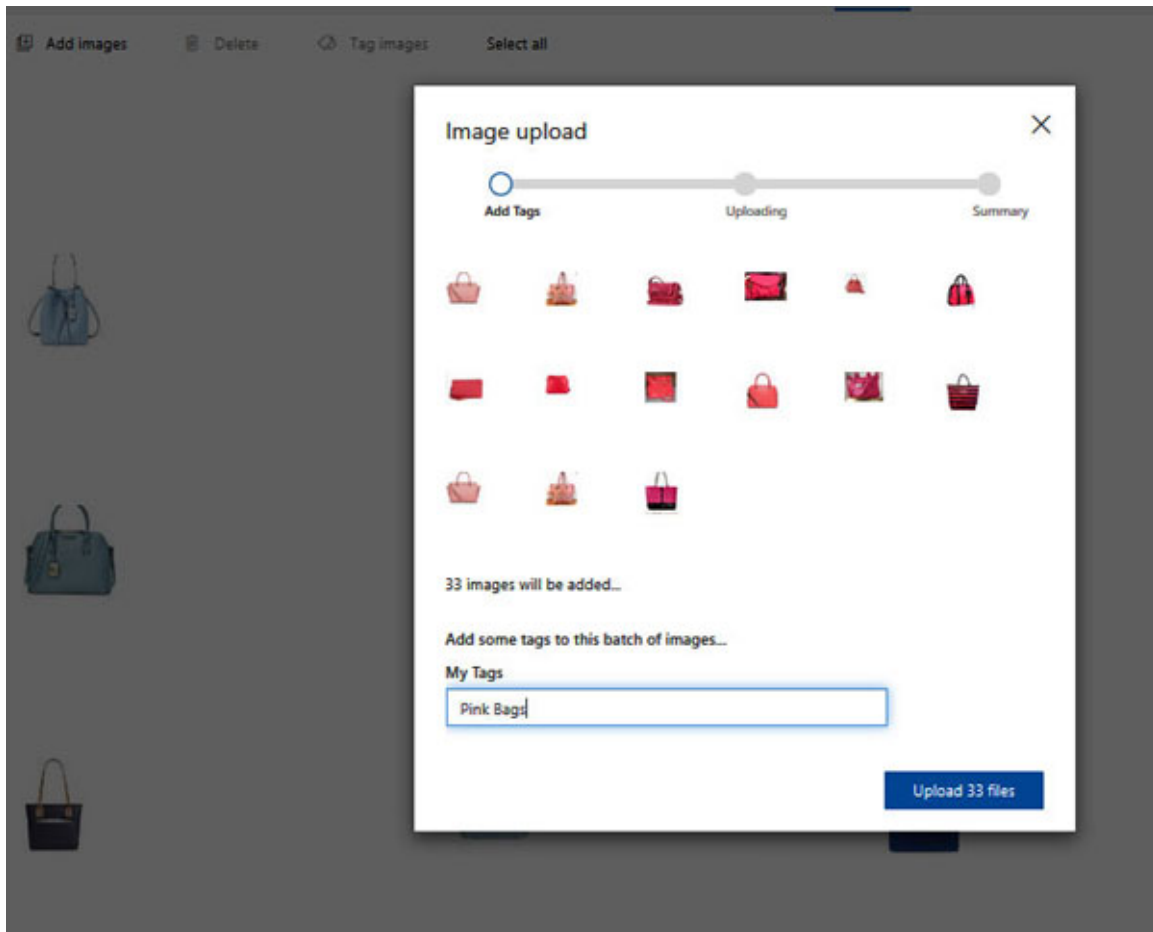
- Name\***: A text input field containing 'AzureAI\_CustomVision'.
- Description**: A text input field containing 'ImageClassificationSample'.
- Resource Group**: A dropdown menu showing 'devNext\_dev [50]' with a 'create new' link to the right.
- Manage Resource Group Permissions**: A link below the resource group dropdown.
- Project Types**: Two radio button options: 'Classification' (selected) and 'Object Detection'.
- Classification Types**: Two radio button options: 'Multilabel (Multiple tags per image)' and 'Multiclass (Single tag per image)' (selected).
- Domains**: A list of radio button options: 'General' (selected), 'Food', 'Landmarks', 'Retail', 'General (compact)', 'Food (compact)', 'Landmarks (compact)', and 'Retail (compact)'.
- Buttons**: 'Cancel' and 'Create project' buttons at the bottom right.

*Figure 2.9: Create New Project*

## Step 2: Upload Images

Once the project is created we can upload images and tag them as required through the interface. A minimum of 50 images will give good accuracy for prediction.

Here we have uploaded ping bags and classified the images as Pink Bags.



*Figure 2.10: Uploading Images*

- Once the images are uploaded with the right tags we need to train the model.

Finished training on 11/29/2020, 4:04:37 PM using **General** domain  
Iteration id: **788bd678-9eb3-4eba-b094-f118b052a2f0**  
Classification type: **Multiclass (Single tag per image)**  
Published as: **Iteration1**



---

*Figure 2.11: Training*

**Step 4: Prediction API**

Once the training of the model is accomplished by selecting Prediction URL, we can leverage the prediction API key, project ID, and the project URL to integrate this into our custom applications.



## How to use the Prediction API



If you have an image URL:

```
https://southcentralus.api.cognitive.microsoft.com/customvision/v3.0/Prediction/861
```

Set **Prediction-Key** Header to : 5c5357a2fd2b4643954dff0f18926010

Set **Content-Type** Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

```
https://southcentralus.api.cognitive.microsoft.com/customvision/v3.0/Prediction/861
```

Set **Prediction-Key** Header to : 5c5357a2fd2b4643954dff0f18926010

Set **Content-Type** Header to : application/octet-stream

Set Body to : <image file>

Got it!

*Figure 2.12: Prediction Key*

### Step 5: Training API

The Custom Vision Service also provides another API called training API. Training API can be leveraged to integrated model building and training into custom applications.

**Step 6:** Create console application and run the sample application: Image Classification using Custom Vision

```
using System;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;

namespace CustomVisionPrediction
{
    class Program
    {
        private static void Main(string[] args)
        {
            //Enter image file path
```

```

        Console.WriteLine("Enter image file path: ");
        string filePath = Console.ReadLine();
        PredictImageOnRequest(filePath).Wait();
        Console.WriteLine("\n\nHit ENTER to exit...");
        Console.ReadLine();
    }
}

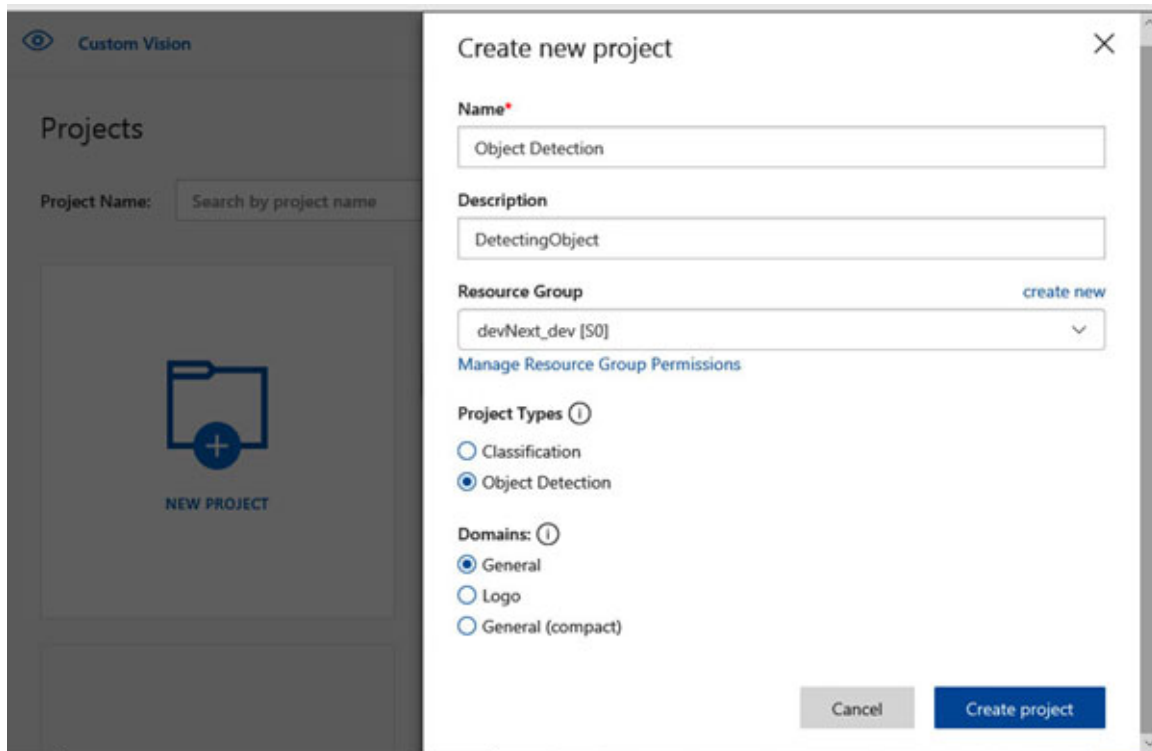
public static async Task PredictImageOnRequest(string _filePath)
{
    // Add valid Prediction Key
    var client = new HttpClient();
    client.DefaultRequestHeaders.Add("Prediction-
    Key", "Replace your Prediction-Key");
    // Replace the below string with the Prediction Url
    string url = "Replace your prediction ";
    HttpResponseMessage responseMessage;
    byte[] byteData = ConvertImageToByteArray(_filePath);
    using (var content = new ByteArrayContent(byteData))
    {
        content.Headers.ContentType = new
        MediaTypeHeaderValue("applicatio
        n/octet-stream");
        responseMessage = await client.PostAsync(url, content);
        Console.WriteLine(await responseMessage.Content
        .ReadAsStringAsync());
    }
}

private static byte[] ConvertImageToByteArray(string filePath)
{
    FileStream fileStreamReader = new FileStream(filePath,
    FileMode.Open,
    FileAccess.Read);
    BinaryReader binReader = new BinaryReader(fileStreamReader);
    return binReader.ReadBytes((int) fileStreamReader.Length);
}
}
}

```

## **Object detection using Azure Custom Vision Service**

We can leverage the Object Detection Project type in Custom Vision Service to detect multiple objects in an image. We can draw boxes around the identified objects.



*Figure 2.13: Using Object Detection*

Once the images are uploaded the model is trained. Once the images are uploaded, objects in the images can be tagged.

We can train the model, test the model, and leverage the model in developing custom applications.

## Conclusion

In this chapter, we have looked at the basics. We have learned how to leverage Azure Vision APIs for image processing, custom vision for image classification, object detection. In the next chapter, we will be exploring optical character recognition, face api, and spatial analysis.

## Multiple choice questions

Thought Experiment - This will be a set of questions around 7-10 to test the knowledge of the concepts learned in the chapter.

There is an audience who has watched a movie. You have the images of the movie and the images of the audience coming out of the theatre.

**a. You want to analyze API using Computer Vision. What is/are the prerequisite/s?**

- a. Install **Microsoft.Azure.CognitiveServices.Vision.ComputerVision** client library
- b. Add the right endpoint for AnalyzeImage API
- c. Add the appropriate subscription key
- d. All the above

**b. Which method you will use to verify two faces belong to the same person?**

- a. Face Detection
- b. Face Recognition
- c. Face Group
- d. Face Verify

**c. How will you build an Image classifier?**

- a. Use Azure Portal
- b. Use Custom Vision
- c. Use Computer Vision website
- d. Use .NET Core based portal

**d. How do you get the Prediction Key for custom vision service?**

- a. Using Azure Portal
- b. Train the model and get the prediction key
- c. Publish the model and get the prediction key
- d. Get Prediction Key from Azure Ops Site

**e. What can we leverage to detect multiple objects in an image?**

- a. Computer Vision
- b. Content Moderation
- c. Detect Object

- d. Scan Content
- f. **What can we leverage to return the domain-specific models supported by Computer Vision API which include celebrity recognizer, landmark recognizer?**
  - a. List Domain Specific Models
  - b. Generate a thumbnail
  - c. Describe Image
  - d. All of the Above
- g. **What is the file size supported by AnalzeImage API?**
  - a. 1 MB
  - b. < 100 MB
  - c. < 4 MB
  - d. < 10 MB
- h. **What method can be used to derive insights on which brands are most popular on social media?**
  - a. Categories
  - b. List Domain-Specific Models
  - c. Detect Brands
  - d. Detect
- i. **What are the domain-specific models supported by the Computer Vision API?**
  - a. Celebrity recognizer
  - b. City Recognizer
  - c. Country Recognizer
  - d. Culture Recognizer
- j. **What method can be used to analyze an image?**
  - a. AnalyseImage
  - b. ExtractImage
  - c. PredictImage
  - d. Detect

## **Answers**

- a. **D. All the Above**
- b. **D. Face Verify**
- c. **B. Use Custom Vision**
- d. **C. Publish the model and get the prediction key**
- e. **C. Detect Object**
- f. **A. List Domain-Specific Models**
- g. **D. < 10 MB**
- h. **C. Detect Brands**
- i. **D. Culture Recognizer**
- j. **A. AnalyseImage**

## **CHAPTER 3**

# **Computer Vision - Optical Character Recognition, Face API, and Spatial Analysis**

**A**fter reading this chapter, the reader will understand the various APIs available in Vision APIs. This includes Text Recognition, Optical Character Recognition, Face API, and Spatial Analysis APIs. By the end of this chapter, the reader will understand how to effectively build applications using each of the APIs.

### **Structure**

In this chapter, the following topics will be explained:

- Text recognition, optical character recognition
- Face API
- Spatial analysis
- Sample use cases

### **Objective**

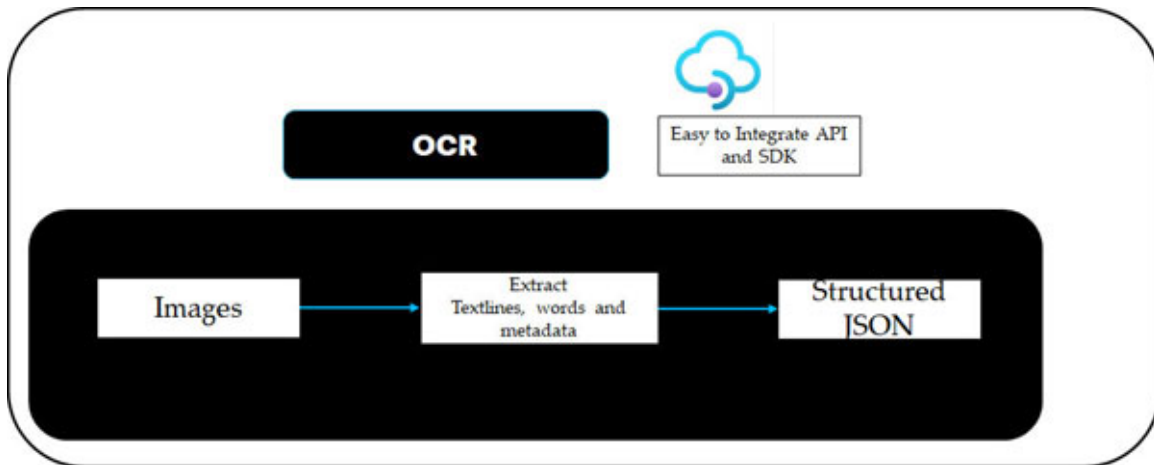
In this chapter, the reader will understand how to use handwriting recognition, optical character recognition, Face API, and spatial analysis. After reading this chapter, the reader will have the skills that make it possible to effectively build applications by leveraging handwriting recognition, optical character recognition, Face API and spatial analysis.

### **Handwritten / Printed text recognition**

Processing handwritten or printed text information is a very important aspect of developing smart applications. Computer Vision provides different APIs which help in detecting and extracting handwritten or printed text in images.

This has many use cases spread across domains such as health, insurance, and banking.

Following diagram shows how we can process printed and handwritten notes using easy-to-integrate APIs/SDKs.



*Figure 3.1: OCR use case*

The following sections cover the different APIs that are available in Computer Vision for OCR.

- **Read API:** Using Read API, we can convert text into a machine-readable character stream. This asynchronous API is good for converting text-heavy images and images that contain a lot of noise. It uses the latest recognition models.

The following are the key features of Read API:

- Supported image types include TIFF, JPEG, BMP, PDF and PNG
  - File size should be less than 20 MB
  - It can process up to 200 pages with a maximum limit of 300 lines per page
- **OCR API:** We can leverage OCR API to convert images into text synchronously. This API tries to detect the orientation of the text and tries to correct the orientation of the image.

The following are OCR API Features:

- Supported image types include BMP, GIF, PNG and JPEG.
- Supports rotation of image by multiples of 90 degrees and small angles of up to 40 degrees. Input image should be greater than or



equal to 40 x 40 and less than or equal to 3200 x 3200 pixels.

- Size should be less than 10 megapixels.
- Size should not be greater than 4 MB.
- OCR supports 25 different languages and can detect the language of the text in the image.

- **Sample Scenarios**

Given below are some sample scenarios/use cases in which the above-mentioned APIs are used.

**Scenario 1:**

- User uploads a cheque.
- The cheque is converted into Digital Text using OCR API.
- The signature is verified.
- The cheque is cleared.

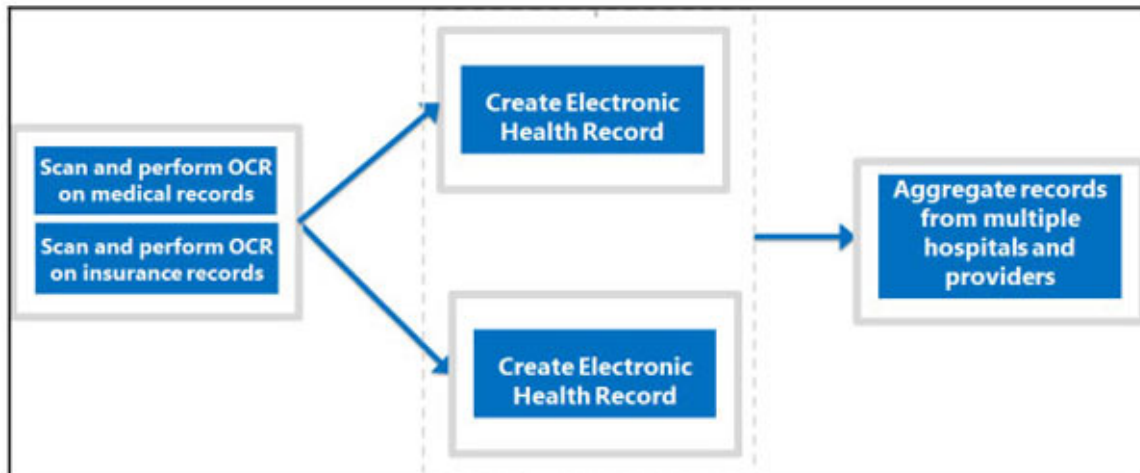


*Figure 3.2: OCR for handwritten cheque use case*

**Scenario 2:** In a scenario in which there are multiple patients' records getting scanned, digitized and stored, providing a comprehensive patient record is very useful for providing apt treatment.

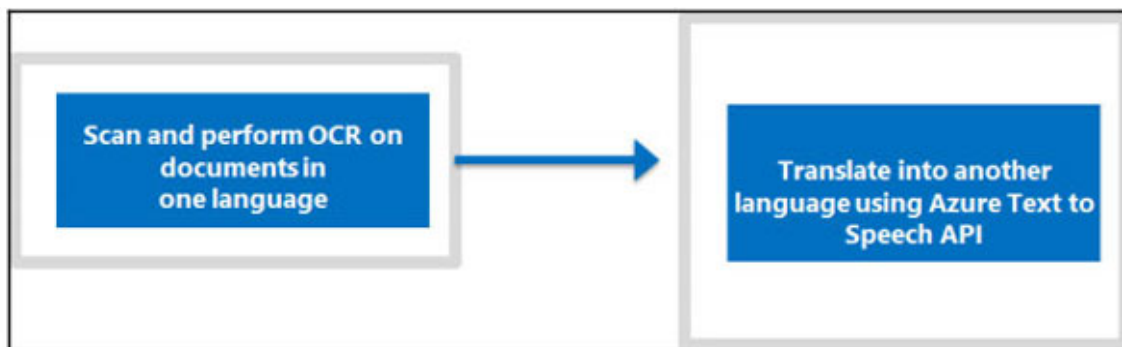
We can create an aggregated patient history by performing OCR on medical records, insurance records from various sources. This provides a comprehensive patient record. This is helpful in providing the right treatment.

Following diagram shows how we can use OCR to aggregate records from multiple sources:



*Figure 3.3: OCR to aggregate records from multiple sources.*

**Scenario 3:** We can scan and perform OCR on documents in one language and translate into another language using Azure Text to Speech API.



*Figure 3.4: OCR to aggregate records from multiple sources.*

## Handwriting recognition lab

Given below is a sample for recognizing handwriting. We will create a sample .NET console application. The application takes a sample image as input. It will recognize and print the text read as the output.

Follow the steps for the handwriting lab:

**Step 1:** Create a new C# .NET Core Console App.

```

using System;
using System.Net.Http.Headers;
using System.Net.Http;
using System.Web;
using System.IO;
  
```

```

using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.Linq;
using Newtonsoft.Json.Linq;

```

**Step 2:** Go to <https://portal.azure.com> and create a computer vision service.

**Step 3:** Replace with your valid subscription key and endpoint.

```

namespace HandwritingDetection
{
    internal class Program
    {
        private static string subscriptionKey = "< subscription key>";
        //Replace subscription key
        private static string
            endpoint =
                "https://retailocr.cognitiveservices.azure.com/vision/v1.0/ocr?"; //Replace endpoint URL
        private static void Main()
        {
            var ch = "1";
            // Get the path and filename to process from the user.
            Console.WriteLine("HandWritten Text Extraction:");
            while (ch == "1")
            {
                Console.Write("Enter the image path with text you wish to read:");
                string imgPath = Console.ReadLine();
                if (File.Exists(imgPath))
                    MakeRequest(imgPath).Wait();
                else
                    Console.WriteLine("\nInvalid file path");
                Console.WriteLine("\nPress Enter 0 to exit and 1 to continue...");
                ch = Console.ReadLine();
            }
        }
    }
}

```

>

#### Step 4: Create a model class **ImageInfoViewModel.cs**

```
namespace HandwritingDetection
{
    public class Word
    {
        public List<int> boundingBox { get; set; }
        public string text { get; set; }
        public double confidence { get; set; }
    }
    public class Line
    {
        public List<int> boundingBox { get; set; }
        public string text { get; set; }
        public List<Word> words { get; set; }
    }
    public class ReadResult
    {
        public int page { get; set; }
        public double angle { get; set; }
        public int width { get; set; }
        public int height { get; set; }
        public string unit { get; set; }
        public string language { get; set; }
        public List<Line> lines { get; set; }
    }
    public class AnalyzeResult
    {
        public string version { get; set; }
        public List<ReadResult> readResults { get; set; }
    }
    public class ImageInfoViewModel
    {
        public string status { get; set; }
        public DateTime createdDateTime { get; set; }
        public DateTime lastUpdatedDateTime { get; set; }
        public AnalyzeResult analyzeResult { get; set; }
    }
}
```

**Step 5:** Add the **MakeRequest()** method and make API call with Request header and Body ;

```
private static async Task<string> MakeRequest(string
imageFilePath)
{
    var errors = new List<string>();
    string extractedResult = "";
    var client = new HttpClient();
    var queryStr = HttpUtility.ParseQueryString(string.Empty);
    // Request headers
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
"YOUR SUBSCRIPTION KEY");
    // Request parameters
    queryStr["language"] = "en";
    var uri = "YOUR CUSTOM VISION SERVICE URL" +
        "/vision/v3.1/read/analyze?" + queryStr;
    HttpResponseMessage httpResponse;
    string opsLocation;
    // Request body
    byte[] byteImageData = GetImageAsByteArray(imageFilePath);
    using (ByteArrayContent content =
new ByteArrayContent(byteImageData))
    {
        content.Headers.ContentType = new
MediaTypeHeaderValue("application/octet-stream");
        // The first REST call starts the async process to analyze
        // the written text in the image.
        httpResponse = await client.PostAsync(uri, content);
    }
    if (httpResponse.IsSuccessStatusCode)
    {
        opsLocation = httpResponse.Headers.GetValues(
            "Operation-Location").FirstOrDefault();
    }
    else
    {
        // Display the JSON error data.
        string error = await httpResponse.Content.ReadAsStringAsync();
        Console.WriteLine("\n\nResponse:\n{0}\n",
```

```

JToken.Parse(error).ToString());
    Console.WriteLine(error);
    return error;
}
string result;
int i = 0;
do
{
    System.Threading.Thread.Sleep(1000);
    httpResponse = await client.GetAsync(opsLocation);
    result = await httpResponse.Content.ReadAsStringAsync();
    ++i;
} while (i < 11 && result.IndexOf("\"status\":\"Succeeded\"") ==
-1);
//If it is success it will execute further process.
if (httpResponse.IsSuccessStatusCode)
{
    var responseData = JsonConvert.DeserializeObject
        <ImageInfoViewModel>(result);
    var linesCount = responseData.analyzeResult.readResults[0]
        .lines.Count;
    for (int j = 0; j < linesCount; j++)
    {
        var imageText = responseData.analyzeResult.readResults[0]
            .lines[j].text;
        extractedResult += imageText + Environment.NewLine;
    }
}
Console.WriteLine("Result:\n-----
\n" + extractedResult + "-----\n");
return extractedResult;
}

```

### Step 6: Add the below method.

```

private static byte[] GetImageAsByteArray(string imgFilePath)
{
    using (FileStream fileReaderStream =
        new FileStream(imgFilePath, FileMode.Open, FileAccess.Read))
    {

```

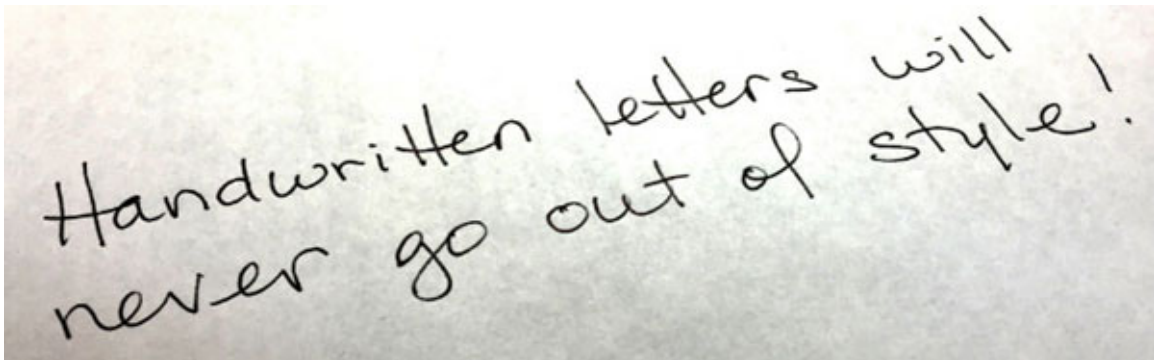
```

        // Read the file's contents into a byte array.
        BinaryReader binReader = new BinaryReader(fileReaderStream);
        return binReader.ReadBytes((int) fileReaderStream.Length);
    }
}
}
}

```

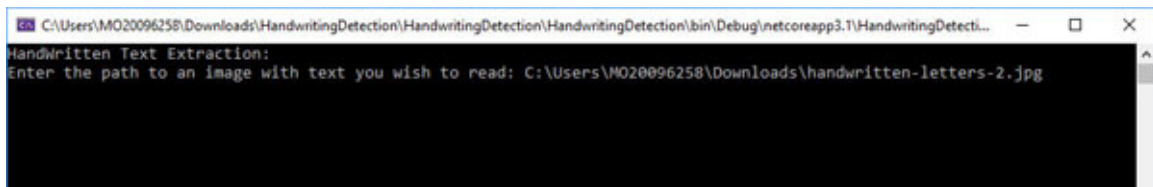
**Step 7:** Build and Run the application.

Sample image:



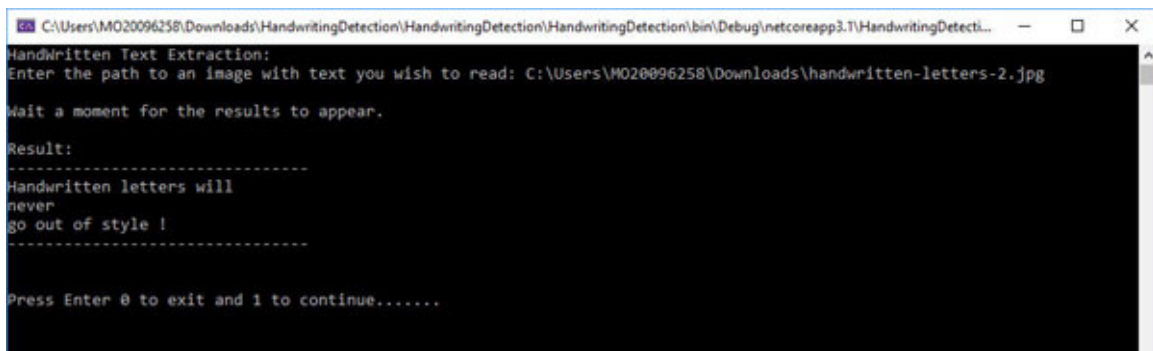
*Figure 3.5: Sample Input*

**Step 8:** Give the image path as an input.



*Figure 3.6: Enter the path of the image*

**Step 9:** Output



*Figure 3.7: Sample output*

## **Face recognition**

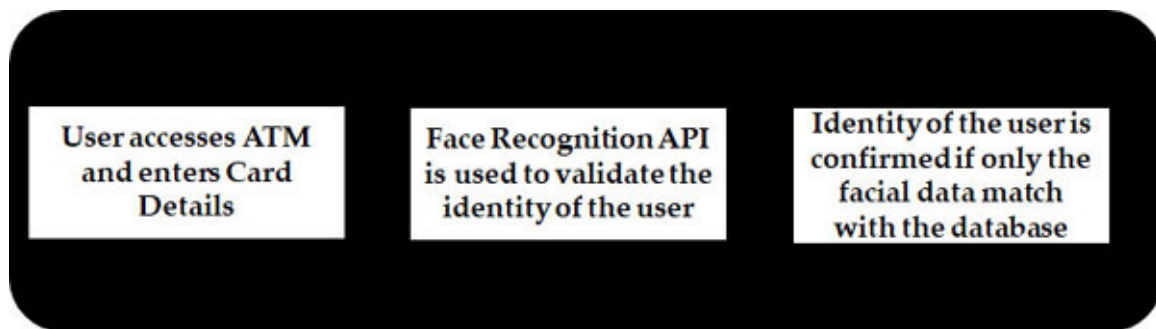
Based on a person's facial contours, facial recognition helps in identifying or verifying a person by comparing and analyzing patterns. Facial recognition technology has potential for a wide range of applications. It makes the person's human machine interactions seamless.

Facial recognition can be effective to solve many problems across industry domains such as banking, healthcare, residential security, and travel.

The Face API provides REST APIs to perform various tasks on faces such as detect faces, verify faces, identify faces, analyze faces, matching faces, analyzing face attributes, grouping similar faces.

We can leverage Face API to detect activities in images, match faces from an existing database of images, return coordinates of faces, manage profiles, analyze and identify faces in video frames.

Following is a sample use case for Face API for Face recognition:

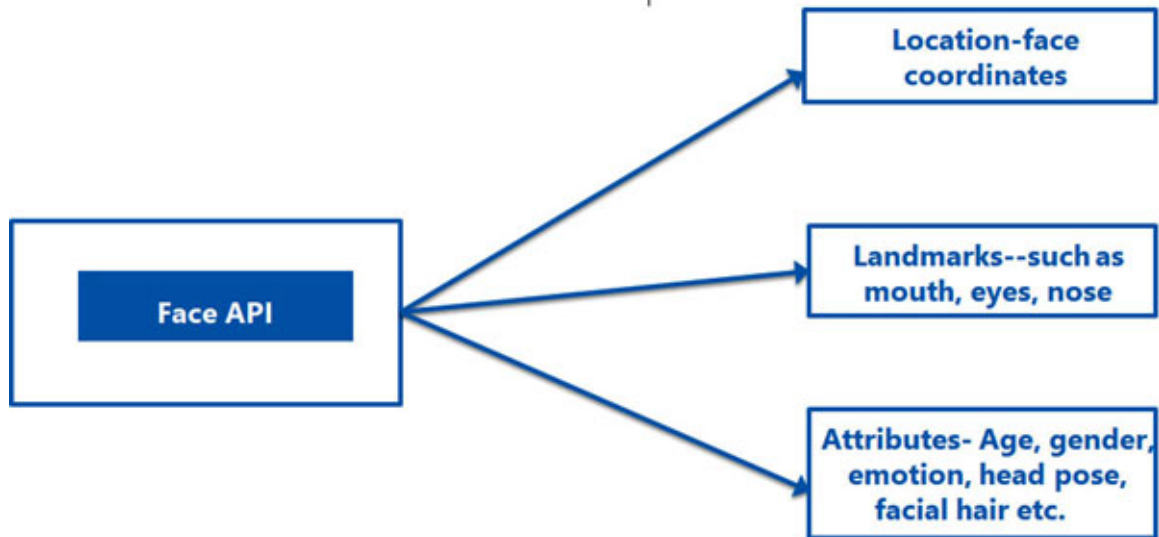


*Figure 3.8: Using Face API for Face recognition*

## **Computer Vision - Face API - In Detail**

The facial data returned by Face API include:





*Figure 3.9: Features provided by Face API*

Some of the use cases include:

- **Identifying Face in a group:** Identify people by comparing faces in a group to return a person object that matches.
- **Identifying Similar Faces:**
  - Match Person applies same person threshold to return similar face.
  - Match Face returns similar faces by ignoring the same-person threshold.

- **Face Detection**

We can leverage Face Detection API to detect faces and return boxes for the images. It also returns facial attributes based on machine learning based predictions. The facial attributes returned include Emotion, Age, Gender, Smile, Facial hair, etc. along with 27 landmarks for each face.

Face – Detect operation helps detect faces in an image, returns face rectangles with faceIds, attributes and landmarks.

Detect human faces in an image, return face rectangles, and optionally with faceIds, landmarks, and attributes.

- Face size  $200 \times 200$  pixels or bigger gives better detection results.
- Supported formats include PNG.GIF, JPEG and BMP format are supported.
- Supported image file size is from 1 KB to 6 MB.

- A maximum of 64 faces can be returned for an image.

## Emotion Recognition

We can leverage the Emotion Recognition API to return a set of emotions for each face in the image with a given level of confidence. Emotions recognized include anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise.

## Face Grouping

We can leverage Face Grouping API to group unidentified faces based on similar features to group them together into groups.

We can use a facelist which could be a single face, a group of faces, a single person or a group of persons to help create and manage groups of faces. It helps find similar faces in a set of pictures which could include celebrities, friends, or family members.

## Computer Vision - Face API Lab

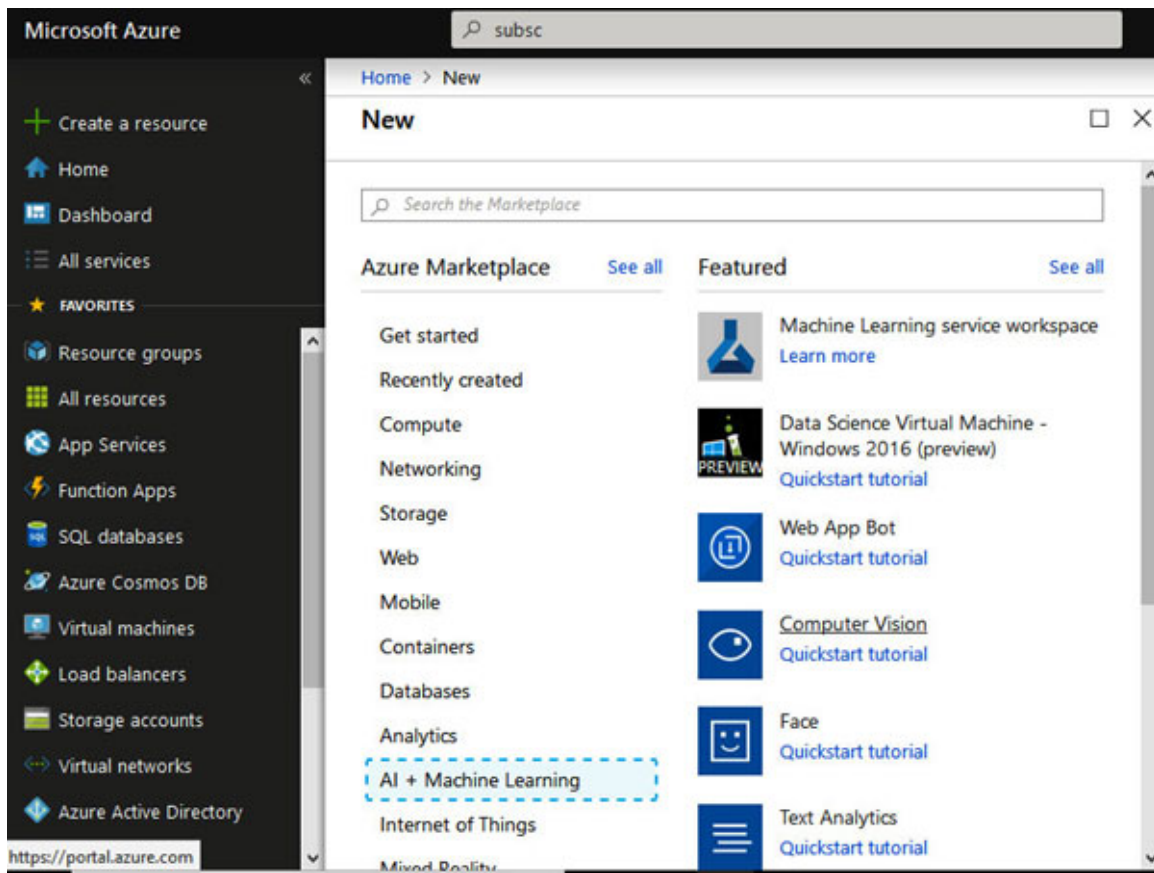
We will create a project using custom vision service and leverage Face API to analyse face.

### Step 1: Prerequisites

- A valid Face API subscription key is required.
- **Visual Studio 2015 or 2017** and above.
- NuGet package:  
**Microsoft.Azure.CognitiveServices.Vision.ComputerVision** client  
library

### Step 2: Create and run the sample application: Face Detection using Face API

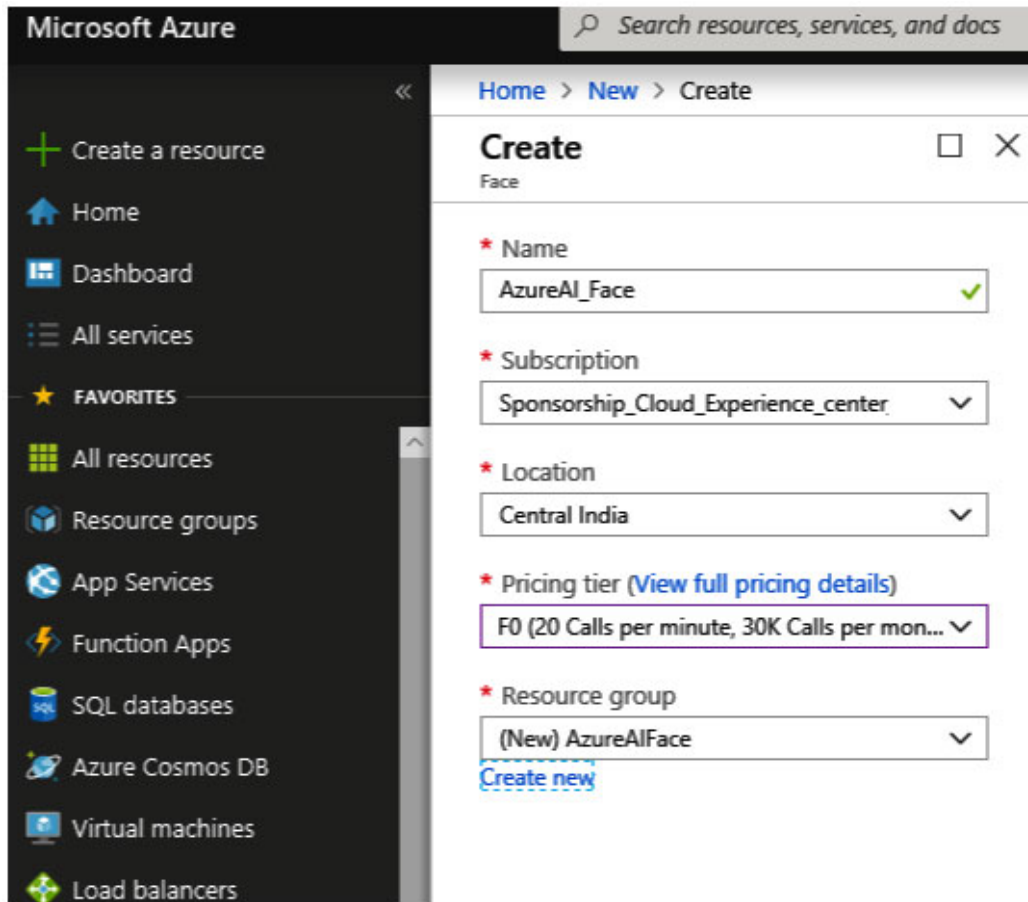
We have to log into the Azure Portal. We have to click on ‘**Create a Resource**’. We have to select ‘**AI + Machine Learning**’. We have to select **Face**.



*Figure 3.10: Create Project*

We have to provide the following details:

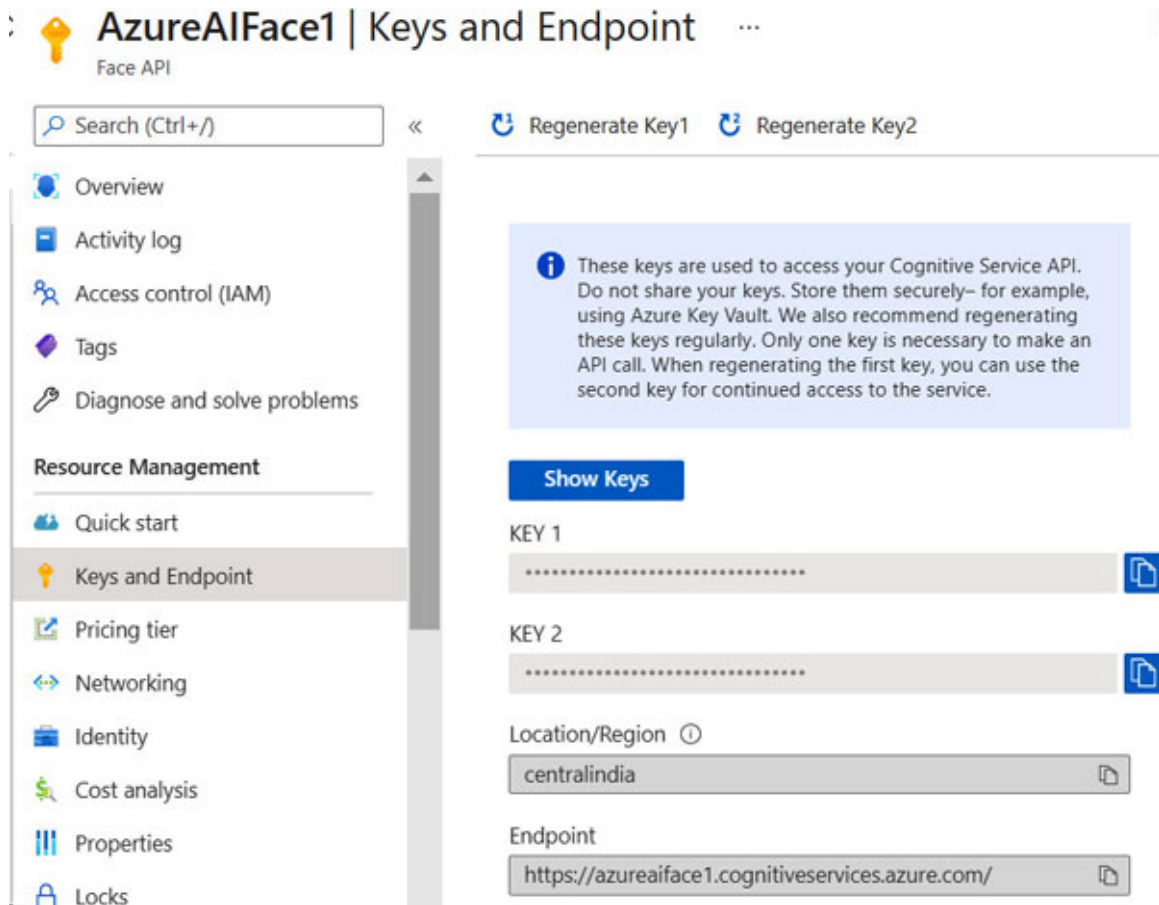
- **Name:** Provide a name for the Face.
- **Subscription:** Select an active subscription
- **Location:** Select the location of the resource group.
- **Pricing Tier:** Select the pricing tier.
- **Resource Group:** Create or select an existing resource group.



*Figure 3.11: Fill Project Details*

**Step 3: Endpoint:** We can choose available locations while creating a Computer Vision API. We can choose the nearest region. We have chosen Central India for this tutorial.

**Step 4: Obtain Subscription Keys:**



*Figure 3.12: Get Subscription Keys*

### Step 5: Create a sample console application to analyze a face:

```
using System;
using System.IO;
using System.Text;
using System.Net.Http;
using System.Net.Http.Headers;
namespace AzureAI_FaceDetection
{
    internal class Program
    {
        // Replace <Subscription Key> with your subscription key.
        private const string faceDetectionSubscriptionKey =
            "<Subscription Key>";
        // replace <myresourcename> with the string found in your
        endpoint URL
        private const string uriBase =
```

```

        "https://azureaifacel.cognitiveservices.azure.com//face/v
        1.0/detect";
private static void Main(string[] args)
{
    Console.WriteLine("Detect Faces: ");
    Console.WriteLine("Enter the path of an image for
    analysis: ");
    string imgFilePath = Console.ReadLine();
    if (File.Exists(imgFilePath))
        FaceAnalysisRequest(imgFilePath);
    else
        Console.WriteLine("\nFile path is Invalid.\nPress Enter
        to exit...\n");
    Console.ReadLine();
}
// Uses Face REST API to Analyze the Image.
private static async void FaceAnalysisRequest(string
imageFilePath)
{
    HttpClient httpClient = new HttpClient();
    // Request headers.
    httpClient.DefaultRequestHeaders.Add(
        "Ocp-Apim-Subscription-Key",
        faceDetectionSubscriptionKey);
    // Request parameters.
    string parameters =
        "returnFaceId=true&returnFaceLandmarks=false" +
        "&returnFaceAttributes=age,gender,headPose,smile,facialHair,glass
es," +
        "emotion,hair,makeup,occlusion,accessories,blur,exposure,noise";
    // Below URI for the REST API Call.
    string requestUri = uriBase + "?" + parameters;
    HttpResponseMessage response;
    // Request body. Posts a locally stored image.
    byte[] imgByteData =
        ConvertImageToByteArray(imageFilePath);
    using (ByteArrayContent content = new
        ByteArrayContent(imgByteData))
    {

```

```

        content.Headers.ContentType =
            new MediaTypeHeaderValue("application/octet-stream");
        // Execute the REST API call.
        response = await httpClient.PostAsync(requestUri,
            content);
        // Get the JSON response.
        string contentString = await
            response.Content.ReadAsStringAsync();
        // Display the JSON response.
        Console.WriteLine("\nResponse:\n");
        Console.WriteLine(JsonPrettyPrint(contentString));
        Console.WriteLine("\nPress Enter key to exit...");
    }
}
// Converts specified file to byte array.
private static byte[] ConvertImageToByteArray(string
    imagePath)
{
    using (FileStream fileStreamReader =
        new FileStream(imageFilePath, FileMode.Open,
            FileAccess.Read))
    {
        BinaryReader binReader = new
            BinaryReader(fileStreamReader);
        return binReader.ReadBytes((int)
            fileStreamReader.Length);
    }
}
// Formats the given JSON string by adding line breaks and
indents.
private static string JsonPrettyPrint(string json)
{
    if (string.IsNullOrEmpty(json))
        return string.Empty;
    json = json.Replace(Environment.NewLine,
        "").Replace("\t", "");
    StringBuilder sb = new StringBuilder();
    bool quote = false;
    bool ignore = false;

```

```

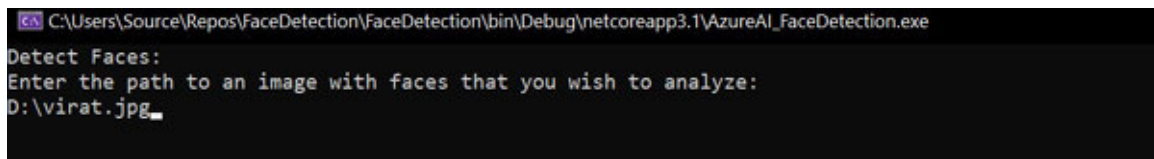
int offset = 0;
int indentLength = 3;
foreach (char c in json)
{
    switch (c)
    {
        case '\"':
            if (!ignore) quote = !quote;
            break;
        case '\\':
            if (quote) ignore = !ignore;
            break;
    }
    if (quote)
        sb.Append(c);
    else
        switch (c)
        {
            case '{':
            case '[':
                sb.Append(c);
                sb.Append(Environment.NewLine);
                sb.Append(new string(' ', ++offset * indentLength));
                break;
            case '}':
            case ']':
                sb.Append(Environment.NewLine);
                sb.Append(new string(' ', --offset * indentLength));
                sb.Append(c);
                break;
            case ',':
                sb.Append(c);
                sb.Append(Environment.NewLine);
                sb.Append(new string(' ', offset * indentLength));
                break;
            case ':':
                sb.Append(c);
                sb.Append(' ');
                break;

```



```
        default:
            if (c != ' ') sb.Append(c);
            break;
    }
}
return sb.ToString().Trim();
}
}
```

The Output is as follows:

A screenshot of a terminal window with a black background and white text. The title bar at the top reads "C:\Users\Source\Repos\FaceDetection\FaceDetection\bin\Debug\netcoreapp3.1\AzureAI\_FaceDetection.exe". The terminal content shows the prompt "Detect Faces:" followed by the instruction "Enter the path to an image with faces that you wish to analyze:". Below this, the user has entered the path "D:\virat.jpg\_" and the cursor is at the end of the line.

```
C:\Users\Source\Repos\FaceDetection\FaceDetection\bin\Debug\netcoreapp3.1\AzureAI_FaceDetection.exe
Detect Faces:
Enter the path to an image with faces that you wish to analyze:
D:\virat.jpg_
```

*Figure 3.13: Sample output*

The output is as follows: It gives the details after analyzing the face.

Wait a moment for the results to appear.

Response:

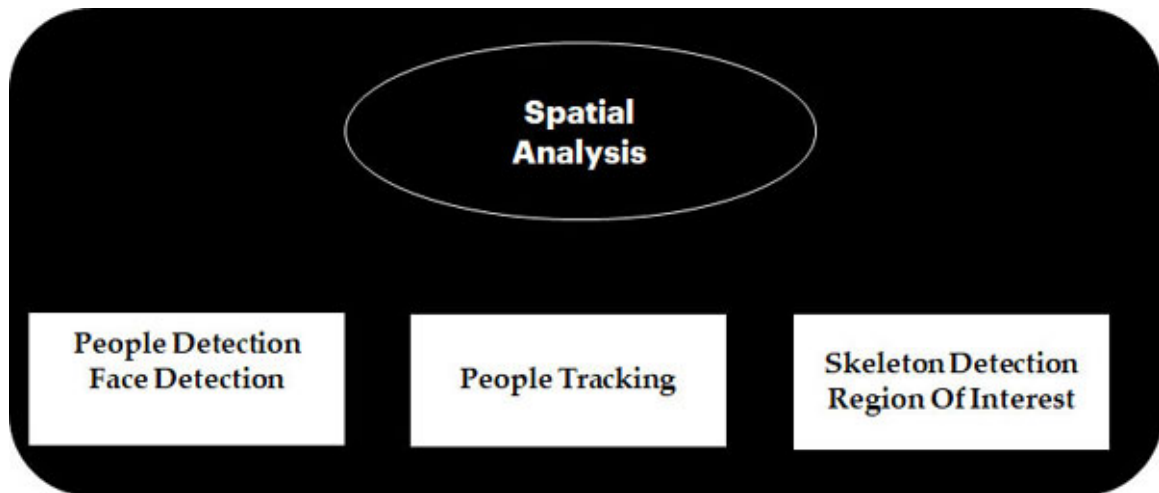
```
[
  {
    "faceId": "0945527f-1d45-459c-996d-77bbce1ff27c",
    "faceRectangle": {
      "top": 56,
      "left": 56,
      "width": 73,
      "height": 73
    },
    "faceAttributes": {
      "smile": 0.0,
      "headPose": {
        "pitch": 12.1,
        "roll": -1.3,
        "yaw": -0.6
      },
      "gender": "male",
      "age": 28.0,
      "facialHair": {
        "moustache": 0.6,
        "beard": 0.6,
        "sideburns": 0.6
      },
      "glasses": "NoGlasses",
      "emotion": {
        "anger": 0.0,
        "contempt": 0.0,
        "disgust": 0.0,
        "fear": 0.0,
        "happiness": 0.0,
        "neutral": 1.0,
        "sadness": 0.0,
        "surprise": 0.0
      },
      "blur": {
        "blurLevel": "low",
        "value": 0.0
      }
    }
  }
]
```

*Figure 3.14: Sample output*

## Spatial Analysis

Computer Vision Spatial Analysis helps in extracting insights, generating events by taking streaming video as input. It helps detect the number of people entering a space and measuring compliance with social distancing guidelines like wearing face masks. By processing space utilization, organizations can optimize their space usage as well.

Given below are the operations supported by Spatial Analysis:



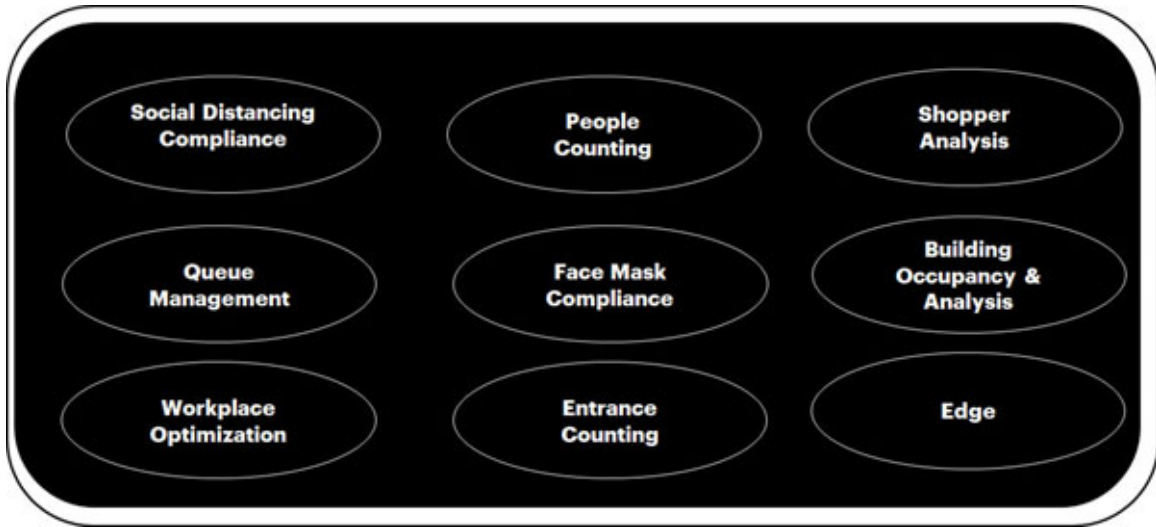
*Figure 3.15: Sample output*

Operation	Usage
People Detection	Helps find people in an image and bounding box coordinates
People Tracking	Helps connect people detections over time as people move around
Skeleton Detection	Helps detect the location of key points of a person's skeleton.
Face Mask Detection	Helps detect the location of a person's face in the camera's field of view and identifies the presence of a face mask.
Region of Interest	It is a user-defined zone or line in the input video frame

*Table 3.1: Spatial Analysis APIs*

## Sample use cases

The following are some use cases which can be achieved using Spatial Analysis



*Figure 3.16: Spatial Analysis use cases*

- **Shopper Analysis:** Helps a grocery store identify how new products are being adopted by understanding the store traffic and measuring the impact of merchandise changes.
- **Queue Management:** Help analyze customer movement in checkout queues and behavior and helps in queue management.
- **Face Mask Compliance:** Helps check if customers are using face masks
- **Building Occupancy & Analysis:** Helps in better space management by analyzing how people are using the workplace.
- **Workplace Optimization:** Helps managers identify employee movement in a restaurant and helps them optimize the same.
- **Entrance Counting:** Helps monitor how long people stay in an area or when they enter through a doorway. Helps in measuring wait times for a checkout line, foot traffic in a lobby, or a specific floor.

## Conclusion

In this chapter, we have looked at the basics of using Azure Vision APIs for handwriting recognition, optical character recognition, Face API, and spatial analysis. In the next chapter, we will learn about Azure AI Learned Services.

## Multiple choice questions

Thought Experiment - This will be a set of questions around 7-10 to test the knowledge of the concepts learned in the chapter.

## **Questions**

- a. **Which API should you use to convert text into a machine-readable character stream asynchronously?**
  - a. OCR API
  - b. Read API
  - c. Detect API
  - d. List API
- b. **Which API should you use to convert text into a machine-readable character stream synchronously?**
  - a. OCR API
  - b. Read API
  - c. Detect API
  - d. List API
- c. **What API helps find people in an image and bounding box coordinates?**
  - a. Find People
  - b. People Detection
  - c. People Find
  - d. Search People
- d. **What is used to check if customers are using face masks**
  - a. A Key Phrase API
  - b. Search People
  - c. Spatial Analysis APIs
  - d. Named Entity Recognition API
- e. **Which API translate into another language using Azure Text to Speech API.**
  - a. Convert Text
  - b. Speech Synthesis
  - c. Speech Recognition API

- d. Text to Speech API
- f. **What can we leverage to identify people by comparing faces in a group to return a person object that matches?**
  - a. Face Compare
  - b. Face Verify
  - c. Face Detect
  - d. Face Similar
- g. **What can we leverage to detect faces and return boxes for the images?**
  - a. Face Compare
  - b. Face Verify
  - c. Face Detect
  - d. Face Similar
- h. **What can be used to return a set of emotions for each face in the image with a given level of confidence? Emotions recognized include anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise**
  - a. Face Verify
  - b. Emotion API
  - c. Happy API
  - d. Face Similar
- i. **What can be leveraged to group unidentified faces based on similar features to group them together into groups?**
  - a. Face Similar
  - b. Face Grouping API
  - c. Emotion API
  - d. Happy API
- j. **Which API detects the number of people entering a space?**
  - a. OCR
  - b. Read

- c. Detect
- d. Spatial Analysis

## **Answers**

- a. **b. Read API**
- b. **a. OCR API**
- c. **b. People Detection**
- d. **c. Spatial Analysis APIs**
- e. **d. Text to Speech API**
- f. **b. Face Verify**
- g. **c. Face Detect**
- h. **b. Emotion API**
- i. **b. Face Grouping API**
- j. **d. Spatial Analysis**

# CHAPTER 4

## Azure Cognitive Services

**A**fter reading this chapter, the reader will understand how to use Decision, Language, Speech and Web Search services of Azure Cognitive Services. The reader will learn about Anomaly detector, Content Moderator, Personalizer, Text Analytics, Azure Functions and Azure Cognitive Service Containers.

### Structure

In this chapter, the topics covered are as follows:

- Decision service
- Content moderator: Use case, Detailed APIs
- Personalizer
- Language service
- Azure Logic Apps and Functions
- Speech service
- Web search service

### Objective

In this chapter, we will explore the decision, language, speech, and Web Search services of Azure Cognitive Services. By the end of this chapter, the reader will understand how to effectively build applications using each of the mentioned Azure Cognitive Services.

### Decision service

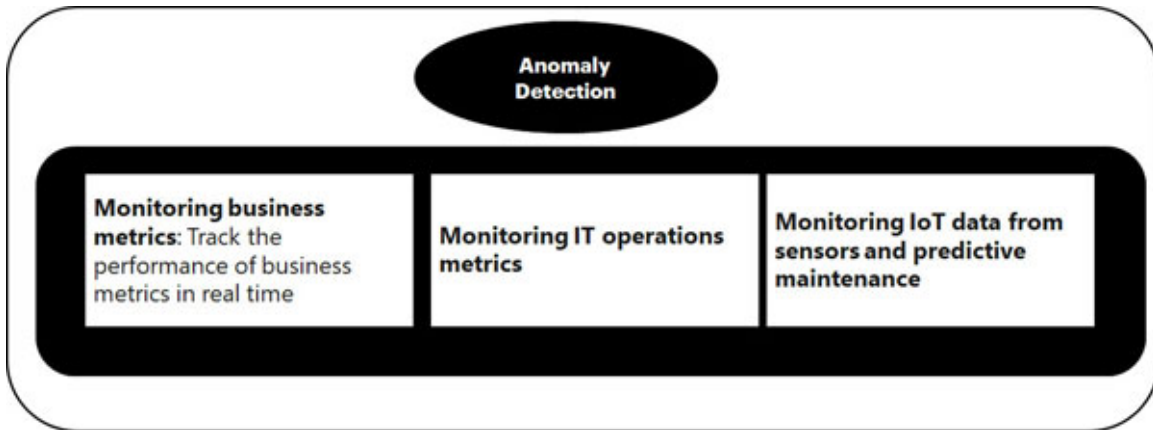
Decision service is used to make good decisions using Decision APIs by helping in analyzing the data and deriving insights from the same.

The main services available under this category are as follows:



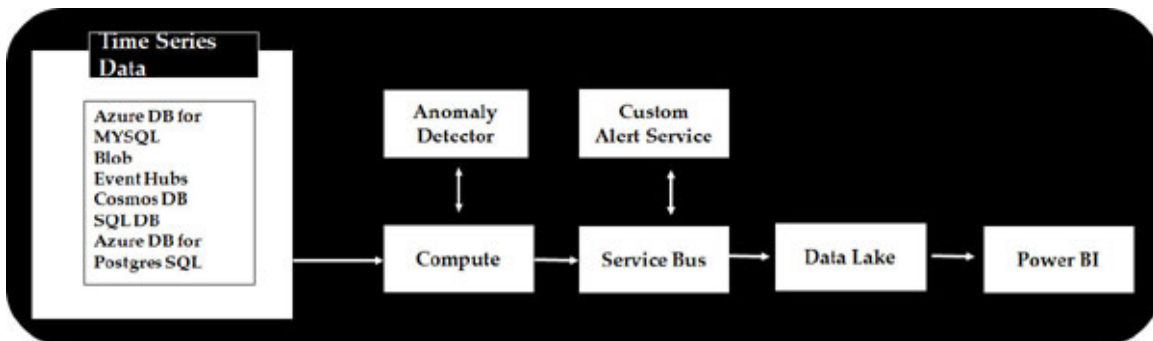
- **Anomaly Detector:** Anomaly detector helps in monitoring and detecting anomalies in the data and can be used in the cloud or edge and can be easily customized. Anomaly detector takes time-series data as input and uses anomaly detection algorithms to help detect dips, spikes in the data.

Some sample use cases where Anomaly detection API can be leveraged are provided as follows:



*Figure 4.1: Anomaly Detection use cases*

We will look at is a sample use case flow for Anomaly Detection:



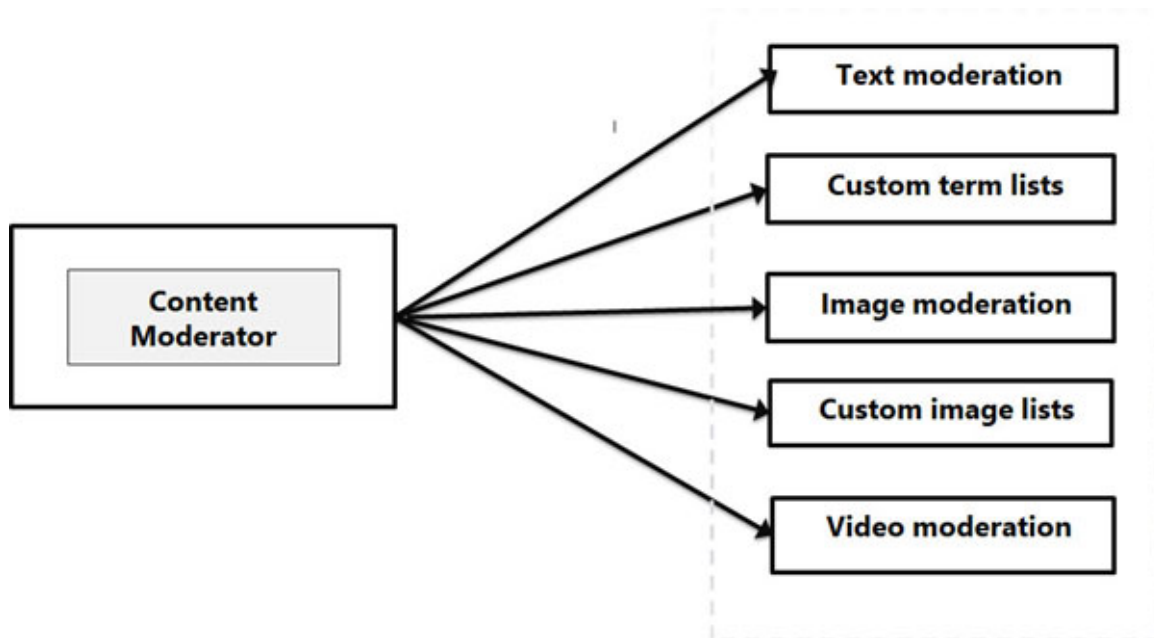
*Figure 4.2: Anomaly Detection Sample Use case Flow*

Time Series data is ingested from data sources which could include SQL DB, Azure DB. Anomaly Service picks the data and detects anomalies. The Anomaly Detector API's algorithms adapt by automatically identifying and applying the best-fitting models to input data, The Anomaly Detector determines the boundaries for anomaly detection as to which are the expected values, and which are anomalies in the data points.

Detailed flow is as follows:

- Data is ingested from data sources like SQL DB, Event Hubs etc.

- Databricks or any other compute resource is used to generate the time series by aggregating and computing the raw data.
- Azure Function picks the message from the message queue based on the anomaly-related metadata and sends the alert about the anomaly.
- Data Lake stores the anomaly detection metadata.
- Power BI is used to visualize the results.
- **Content Moderator:** Content Moderator API helps in monitoring social media applications for offensive content and identifies offensive content like posts or videos and flags those.
- Few scenarios where it can be effectively used include moderating content in social messaging platforms, organizations, schools etc.
- The following figure shows the available APIs that help in moderation of text, images, and videos. We can leverage the APIs to check objectionable content in the text/images.



*Figure 4.3: Content Moderator APIs*

We will look at a sample use case for content moderator.

## [Content Moderator-Use case](#)

When a user uploads an image the program checks if the image has objectionable content.

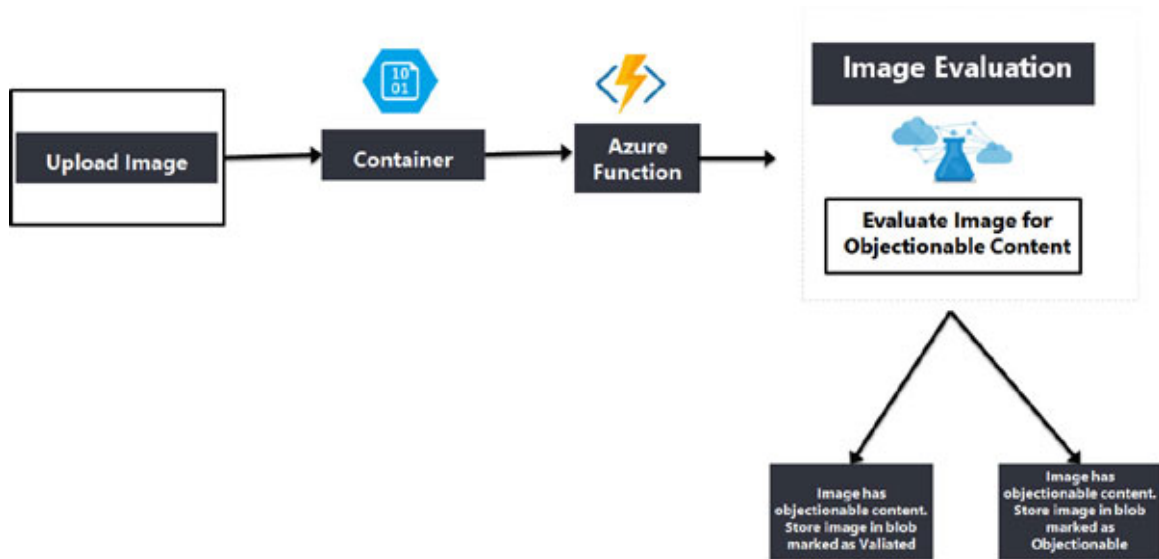
To check whether an image contains objectionable content we will leverage Azure Container, Blob storage, Image Moderation API and Perform the following steps:

The input to be provided is an image that the user uploads.

Given below is the program flow explained in detail:

- User uploads an image.
- The image is stored in a blob.
- Azure function gets triggered when an image is added to the blob.
- Image Moderation API is invoked to check if the image contains racy content.
- Based on the value the image is stored in Validated Blob or Objectionable Blob for further processing.

We can leverage Azure Container, Blob storage, Image Moderation API as given as follows:



*Figure 4.4: Workflow to be developed for the use case*

## Content Moderator-Detailed APIs

Azure Content Moderator cognitive service verifies and flags text, image, and video content for material that is offensive with labels. The Content Moderator service includes Moderation APIs which help in detecting and flagging offensive content.

The following are the details of Custom Image lists and Review APIs:

API group	Usage
<b>Text moderation</b>	Helps scan text for offensive content. Text is classified into three categories – Category 1, 2, and 3 based on the adult content present in the text.
<b>Custom term lists</b>	Helps use custom lists for flagging offensive content.
<b>Image moderation</b>	Helps scan images and identify offensive content.
<b>Custom image lists</b>	Helps scan images against a custom list of images. Custom Image lists: Custom image lists help prevent sharing of offensive images by creating a custom list of offensive images.
<b>Video moderation</b>	Helps scan videos for offensive content.

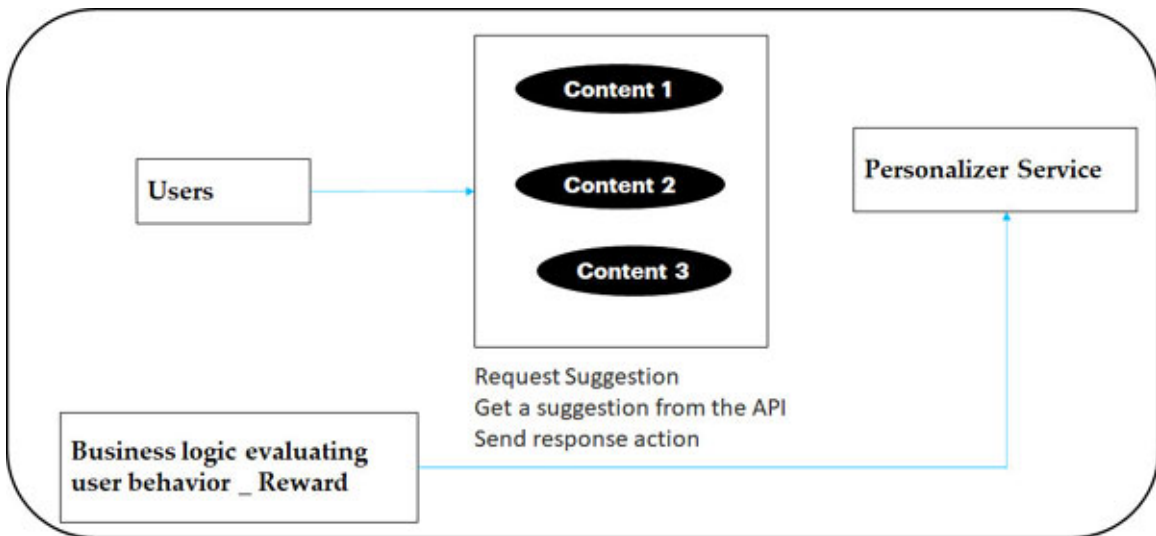
*Table 4.1: Content Moderator-detailed APIs*

## Personalizer

The Personalizer Service is used to create personalized recommendations for users by learning from user-application interactions, based on their online activities.

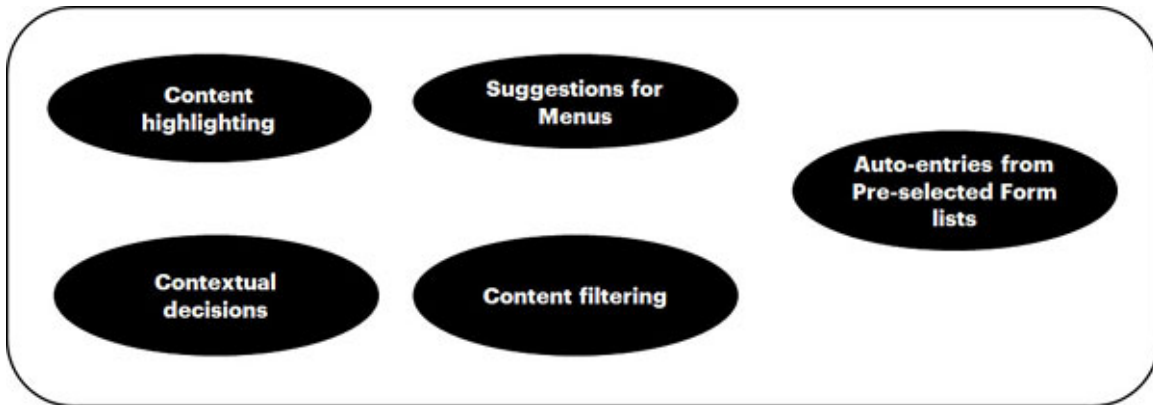
Contextual content is provided to users leveraging reinforcement learning. It uses user behaviors and activities to surface the right content to the users.

It uses reinforcement learning as given as follows:



*Figure 4.5: Personalizer Service*

Some of the use cases for Personalizer Service include highlighting content, providing menu suggestions, filling up form lists for users, providing contextual decisions and filtering content as shown as follows:



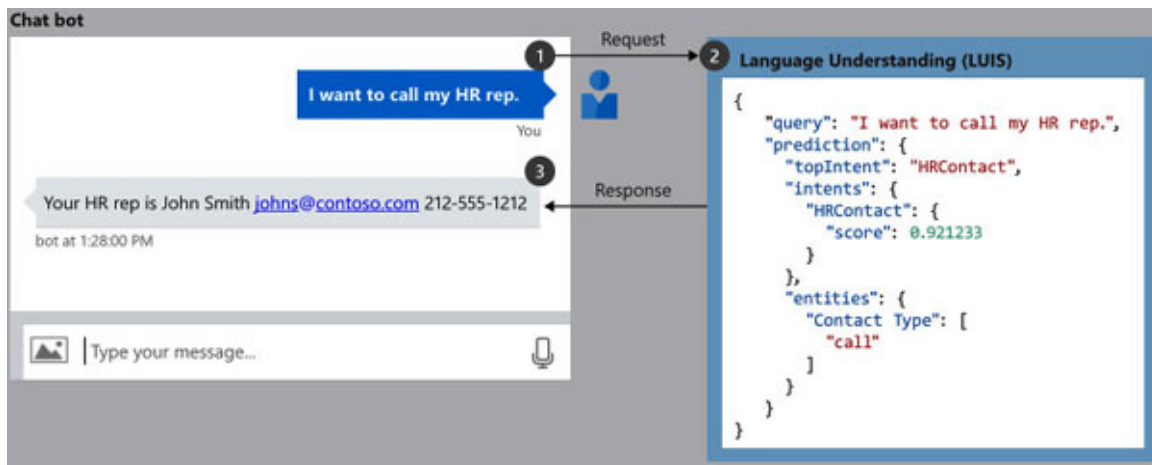
*Figure 4.6: Personalizer Use Cases*

## Language Service

Language APIs help users to analyze texts and recognize intents and entities from text.

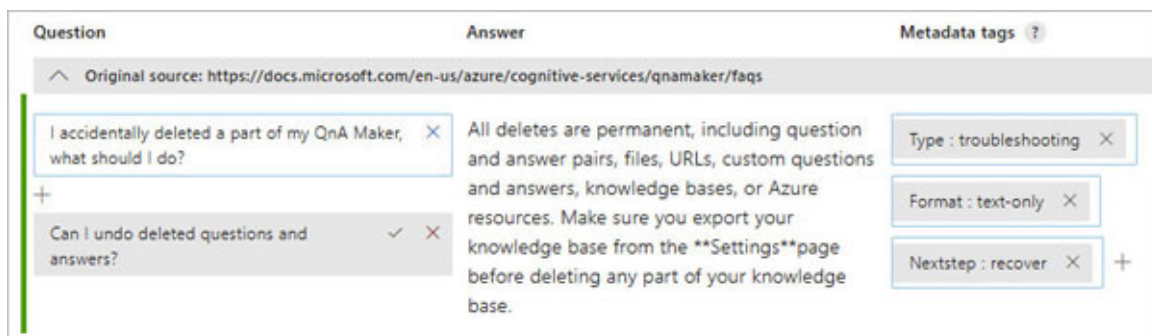
There are several services available under this category as given as follows:

- **Immersive Reader:** Immersive Reader is a standalone web application provided by Microsoft that enables processing the content for parts of speech, text to speech, translation, and more. It implements proven techniques to improve reading comprehension for new readers, language learners, and people with learning differences such as dyslexia.
- **LUIS:** This service provides Natural Language Understanding and Interpretation Services. LUIS is a cloud-based conversational AI service that applies machine-learning intelligence to a user's conversation to understand the intent of the user, and pull out relevant, detailed information. Chatbots can leverage LUIS to understand the intent and provide more contextual responses.
- **QnAMaker:** It is a cloud-based **Natural Language Processing (NLP)** service that allows users to create a natural conversational layer. The QnAMaker can be leveraged to answer questions based on FAQs (custom knowledge base) for chatbots.
- **Translator:** Translator can be leveraged to translate language from one language to another in real-time with support for more than 90 languages.



**Figure 4.7:** Anomaly Detection use cases  
– Language Service – LUIS (Source)

In the above figure, LUIS helps the chatbot to identify the intent and entities linked to the conversation with the user.



**Figure 4.8:** Language Service – QnA Maker

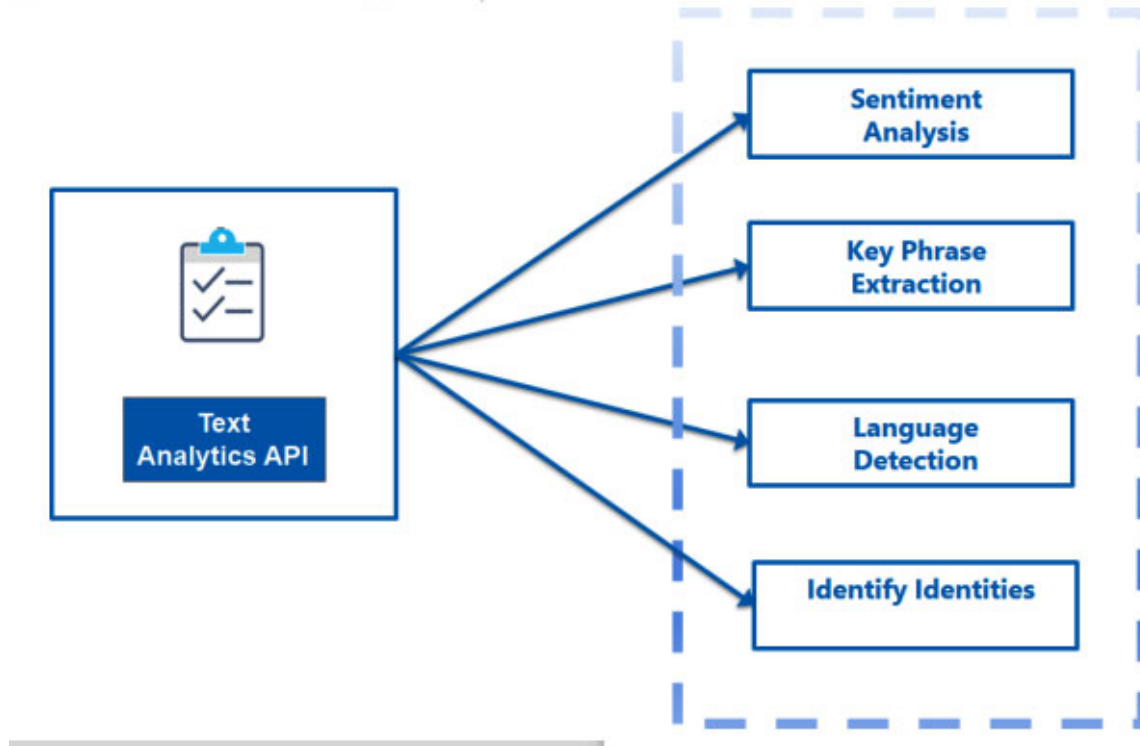
(-<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/media/qnamake>)

This is an example of the QnA Maker that is leveraged to create a FAQ chatbot that can answer users' questions based on the inputs available.

Now that we have a brief overview of the services, we will go into Text Analytics API in more detail.

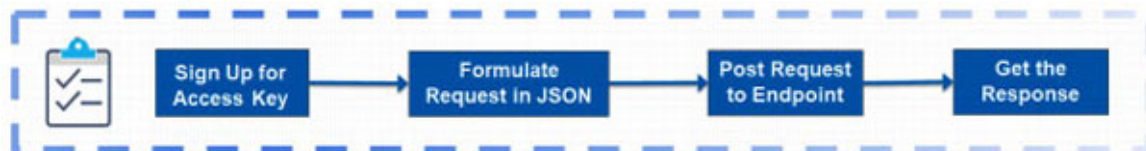
## Text Analytics

Text Analytics API is part of Language API. It provides Natural Language-based Processing APIs to process raw text. We can use Text Analytics in conjunction with Handwriting, OCR API to build smart business applications. It provides the following functionalities



**Figure 4.9:** Text Analytics API.

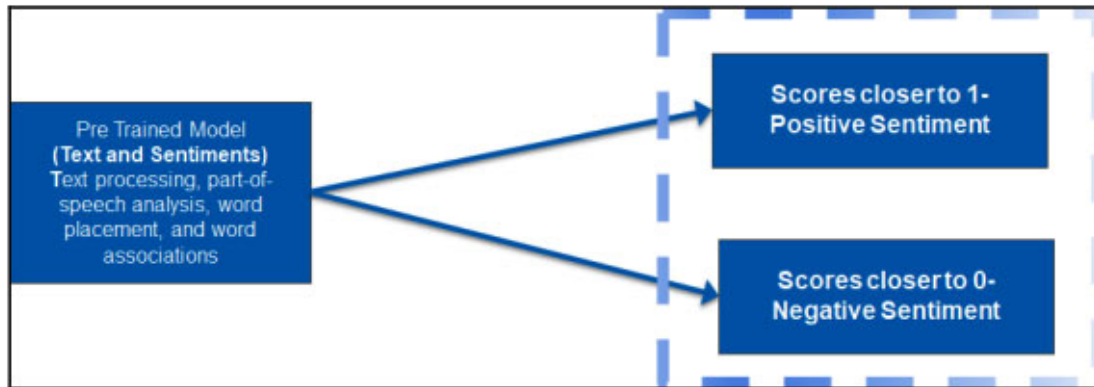
The Text Analytics API can be invoked as follows:



**Figure 4.10:** Text Analytics API Invocation

We can use Sentiment Analysis API to perform sentiment analysis on text and key phrases in discussion forums, reviews and social media. Languages that are supported currently include English, Spanish, German and French.



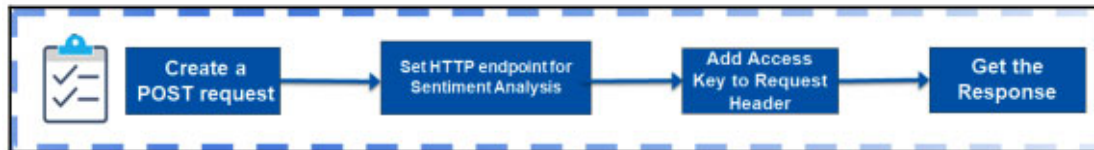


*Figure 4.11: Sentiment Analysis API.*

Following diagram shows how we can use text and sentiment analysis to check whether it is a positive sentiment or a negative sentiment.

Invoking Sentiment Analysis API:

- Input should be provided in the form of JSON documents.
- Document size must be under 5,120 characters per document.
- Limited 1,000 items (IDs) per collection.



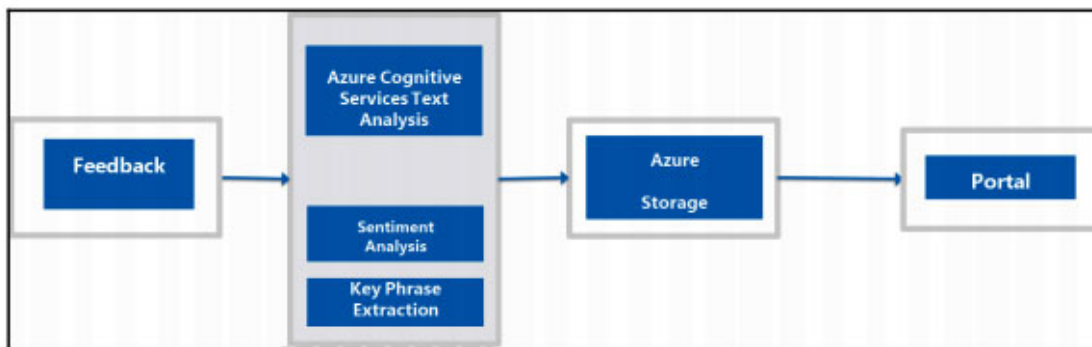
*Figure 4.12: Invoking Sentiment Analysis API.*

1. **Key Phrase Extraction API:** We can use Key Phrase Extraction API to extract a list of strings denoting the key talking points in the input text. The service uses Microsoft Office's sophisticated Natural Language Processing toolkit. Languages supported include Spanish, German, English and Japanese.
2. **Language Detection API:** The Language Detection API takes text documents as input and returns language identifiers. Number of languages supported is 120. This capability is useful in scenarios in which the language is unknown.
3. **Entity Identification API:** The Named Entity Recognition API takes a JSON document and returns a list of entities with links to more information on the web.



**Use Case Scenarios:** The below are sample use case scenarios in which Azure Cognitive Text Analytics is combined with Sentiment Analysis, Key Phrase Extraction API and other APIs.

- **Analyze Feedback results:** Analyze the feedback from a session etc. using Azure Cognitive Services Text Analytics. The following shows how we can analyze feedback and store in Azure storage and display reports in the portal.



*Figure 4.13: Analyze Feedback results.*

- **Process and categorize support incidents:** Using Key Phrase Extraction and Entity Recognition we can process requests to route the requests to the right support engineers. We can also automate trend analysis which will help in the right planning of resources for resolving the incidents.

We will look into a brief introduction of Azure Logic Apps and Functions before getting into a sample use case.

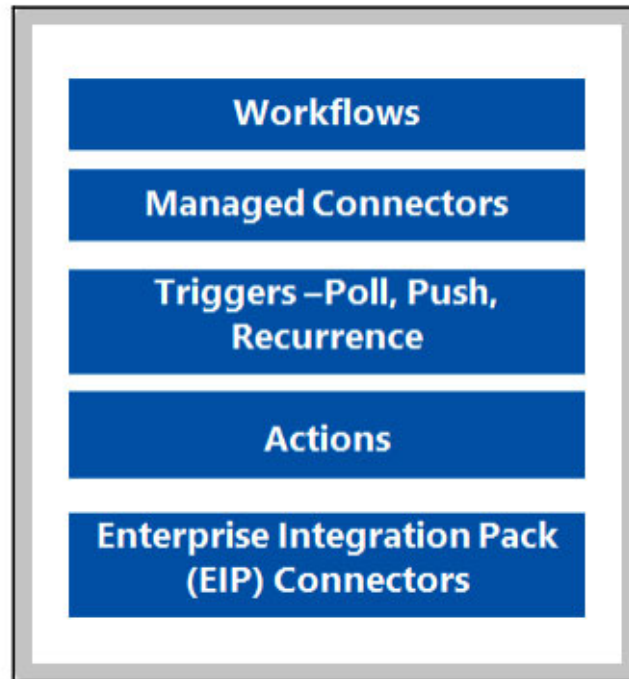
## [Azure Logic Apps and Functions](#)

Let us have a brief introduction to Azure Logic Apps and Azure Functions which we will leverage in executing the lab.

- **Azure Logic Apps - A Brief Overview**

Azure Logic Apps are serverless and scalable applications that are used to perform workflows and integration.

Characteristics of Azure Logic Apps are shown below:



*Figure 4.14: Analyze Feedback results.*

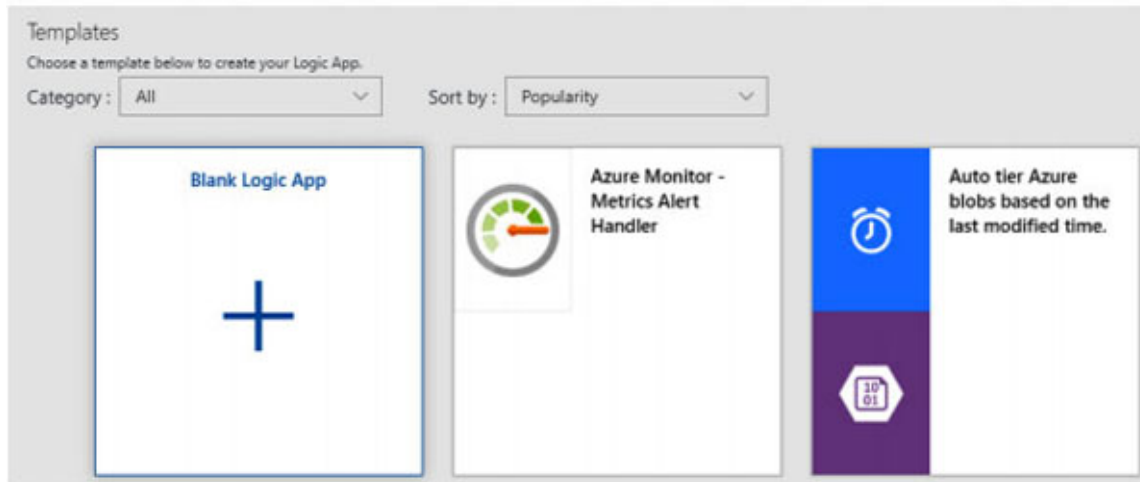
Logic Apps can be used to define workflows. They come with a set of built in connectors to connect a number of data sources and services such as FTP. They also come with a set of Enterprise Integration Pack connectors such as BizTalk. These connectors can be used for complex enterprise integration scenarios which include validation and XML messaging and industry-standard protocols, including AS2, X12, and EDIFACT. The prerequisite for using Enterprise Integration Pack is an “*Integration Account*”.

They provide a number of triggers for initiating the workflow, including:

- **Poll Trigger:** Poll at regular intervals to check for an event or data.
- **Push Trigger:** Listen to an endpoint or an event.
- **Recurrence Trigger:** Triggers the workflow at prescribed Actions allow operations in a workflow.

**Templates:** Logic Apps come with a set of pre-defined templates and support for continuous integration and deployment.

**Azure Functions:** Azure Functions are serverless and are priced based on their usage. They are invoked by Azure Functions Runtime on the advent of an event. Functions are pieces of code that are written in any language and are invoked by triggers. Characteristics of Azure Functions are shown as follows:



*Figure 4.15: Create Azure Function*

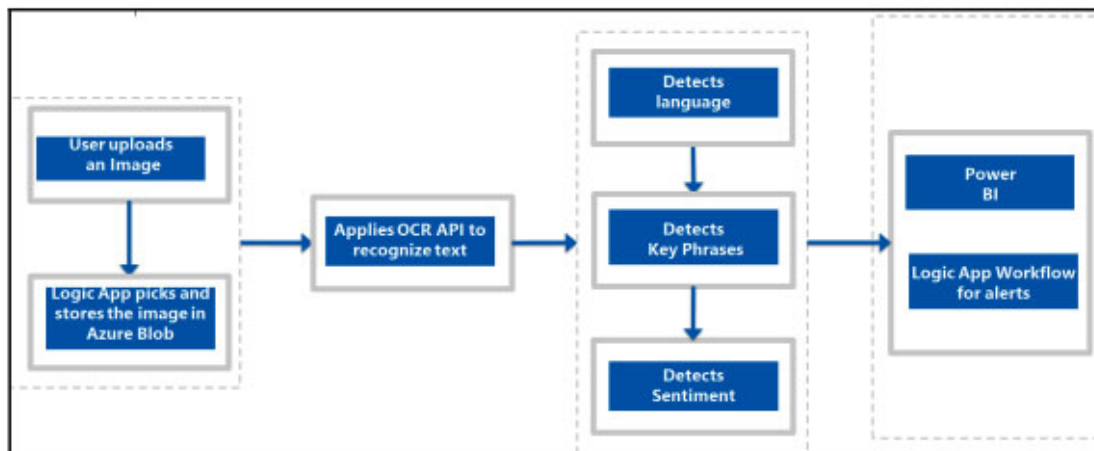
Triggers available in Azure Functions include:

- **HTTP:** RESTful HTTP endpoints are triggered when an HTTP endpoint is requested.
- **Timer:** Triggered on a predefined schedule.
- **WebHook:** Triggered when GitHub is consumed.
- **CosmosDb:** Triggered on changes in CosmosDB.
- **Blob:** Triggered when a file is uploaded in Azure Blob.
- **Storage Queue:** Triggered when a message is added or removed from Azure Queue Storage.
- **EventHub:** Triggered when a registered Azure Hub Event occurs.  
Service Bus: Triggered when a Service Bus topic is updated.
- **Bindings:** Are optional and help simplify coding for data input and output.

## **Lab - Create Logic App using OCR and Text Analytics and Azure Functions**

We will create a simple application which enables a field worker to upload data he has collected offline. Once the image is uploaded the image is stored in Azure Blob Storage. A logic app that is listening to the event picks up the image and runs a workflow, the functionalities of which include applying OCR, detecting language, detecting key phrases and storing the results. A

Power BI app shows reports as required. The following is the sample scenario addressed by the lab:



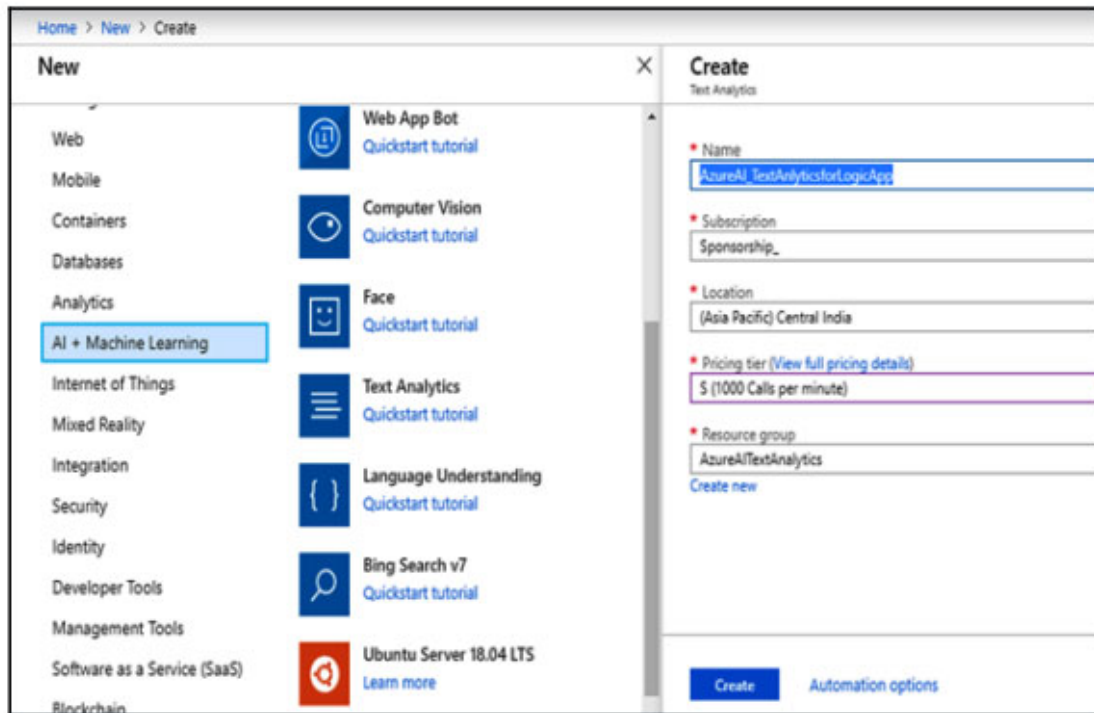
*Figure 4.16: Use case Flow Diagram*

Shown below are the detailed steps for performing the experiment for the problem statement given above:

**Step 1:** We will obtain Cognitive Services API subscription for Text Analytics API as given below:

Go to Azure portal and Select '**AI + Machine Learning**' -> **Text Analytics**

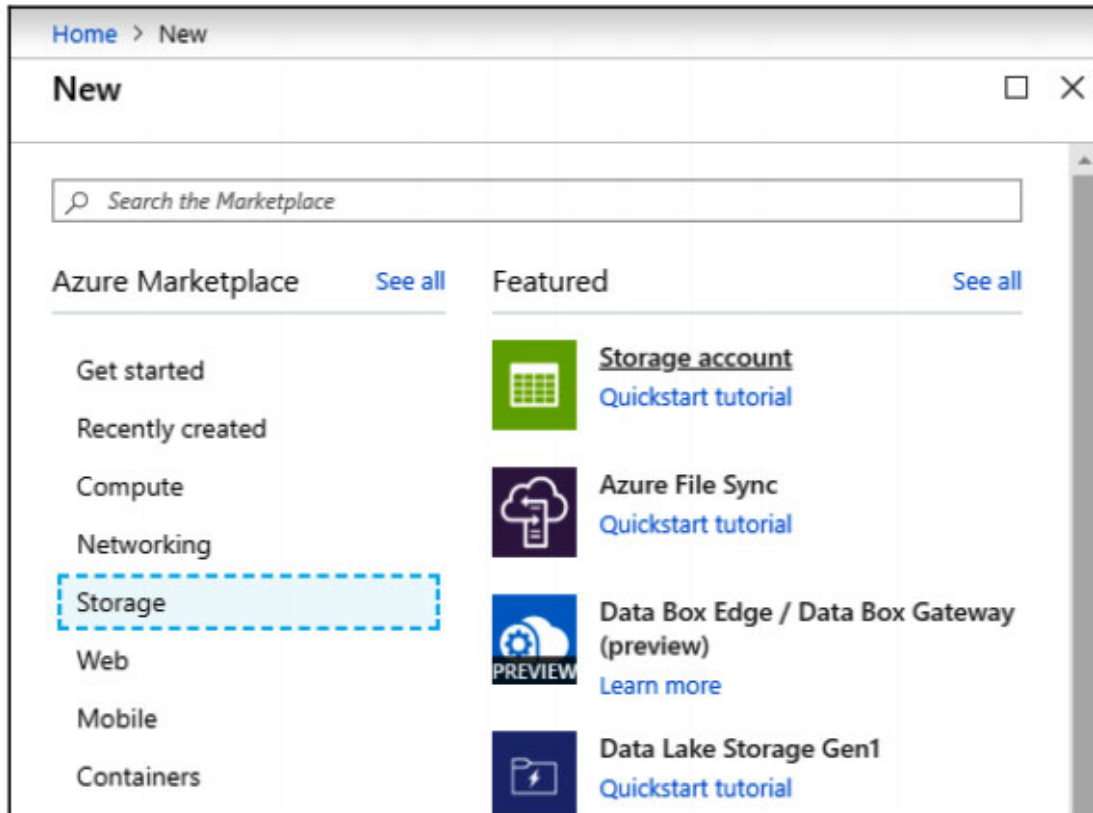
**Step 2:** We will create a Cognitive Service API subscription for Text Analytics API.



*Figure 4.17: Create Subscription Key*

**Step 3:** We will obtain an OCR subscription key as described in the Handwriting Recognition Lab.

**Step 4:** We will create a storage account.



*Figure 4.18: Create Storage Account*

Once Validation is passed, select **Create**.

**Create storage account**

✓ Validation passed

Basics Advanced Tags Review + create

**BASICS**

Subscription	
Resource group	AzureAITextAnalytics
Location	(Asia Pacific) Central India
Storage account name	azureaiblobstorage
Deployment model	Resource manager
Account kind	StorageV2 (general purpose v2)
Replication	Read-access geo-redundant storage (RA-GRS)
Performance	Standard
Access tier (default)	Hot

**ADVANCED**

Secure transfer required	Enabled
Allow access from	All networks
Hierarchical namespace	Disabled

Create Previous Next [Download a template for automation](#)

*Figure 4.19: Create Storage Account*

**Step 5:** We will create a Container using the storage account.

**Step 6:** We configure the Blob Storage as outlined below.

- Search Blob Storage
- Choose the Storage Connection
- Next, Choose the folder path, Blob name, and Blob content
- We will create a Container using the storage account

**Step 7:** We will create a logic app. In the dashboard, choose.

- Create a resource
- Select Integration
- Select Logic App

We will add details for the Logic App.

Home > New > Logic App

### Logic App

Create

\* Name  
AzureAllLogicApp ✓

\* Subscription  
[Dropdown]

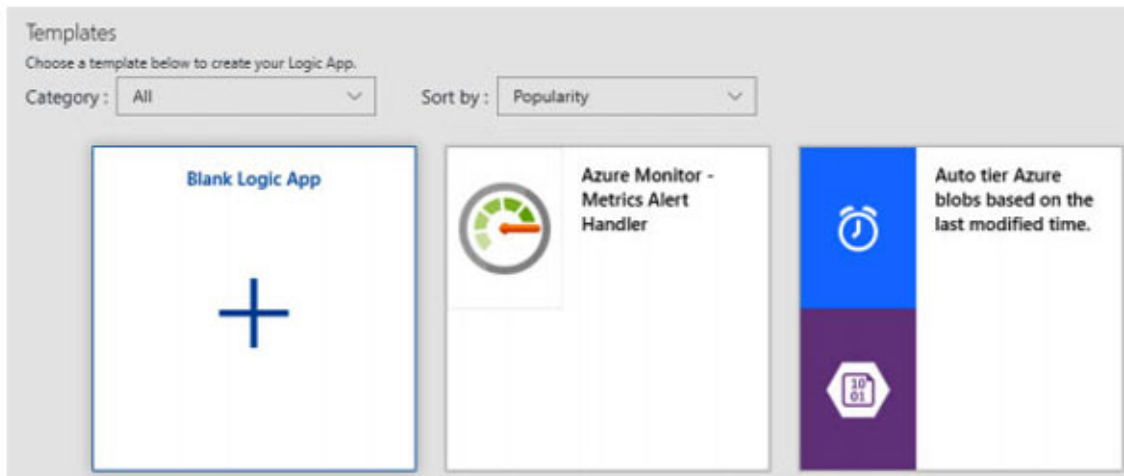
\* Resource group ⓘ  
☒ Create new ☐ Use existing  
[Text Box]

\* Location  
Southeast Asia [Dropdown]

Log Analytics ⓘ

**Figure 4.20:** Add Details for Logic App

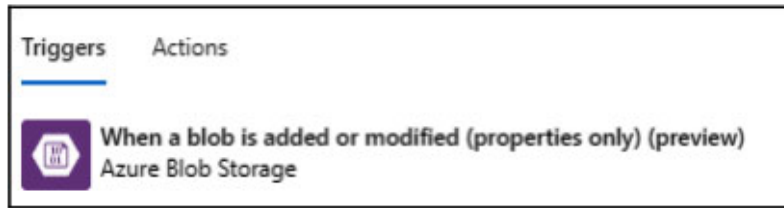
We will now add a Trigger for when a blob is added or modified. Shown below is the way we will do it.



**Figure 4.21:** Add Trigger

We will add interval and frequency details.





*Figure 4.22: Set the interval and frequency for the Trigger*

**Step 8:** Configuring Computer Vision API for OCR: We will now configure the computer Vision OCR API for detecting the sentiment of the message.

Search for Computer Vision

When a blob is added or modified (properties only) (Preview) ...

Computer Vision API

\* Connection Name  
ComputerVisionConnection

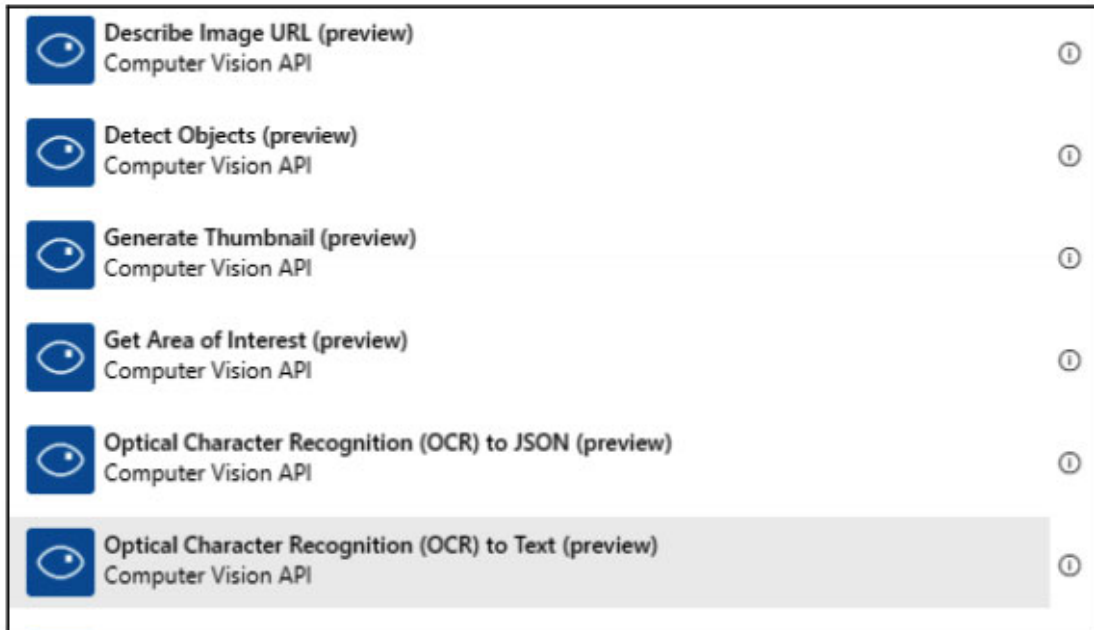
\* Account Key  
\*\*\*\*\*

Site URL  
https://centralindia.api.cognitive.microsoft.com/text/analytics/v2.0

Create

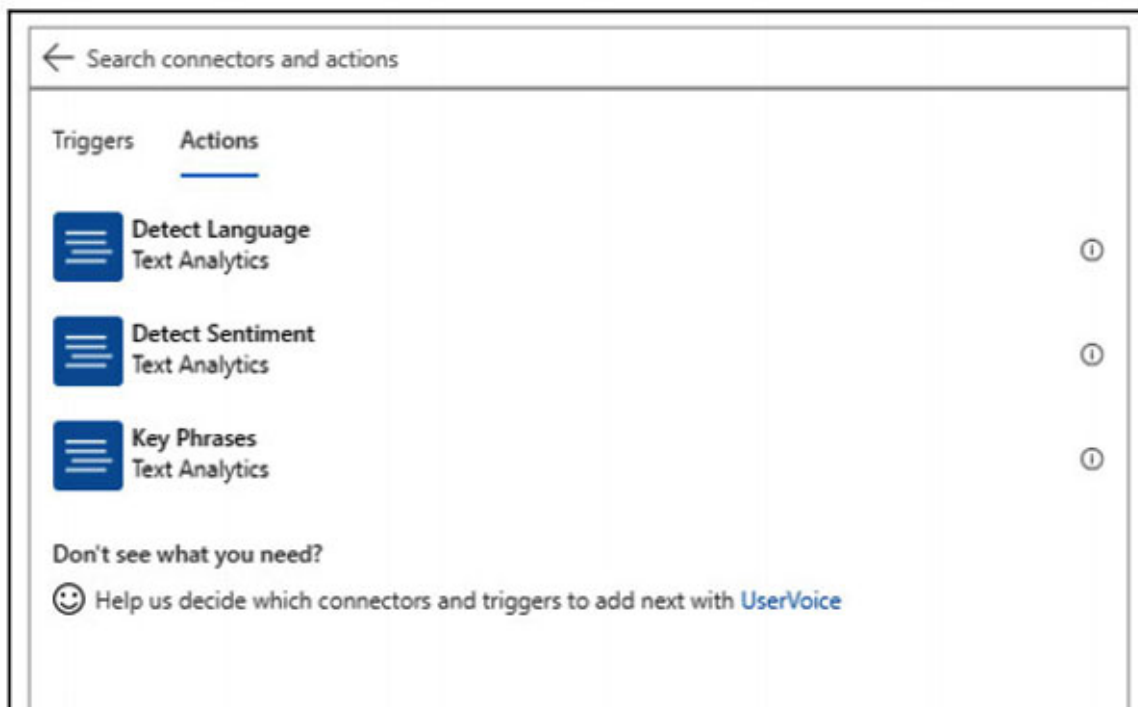
*Figure 4.23: Configure Computer Vision API*

Select '**Optical Character Recognition (OCR)** to Text.



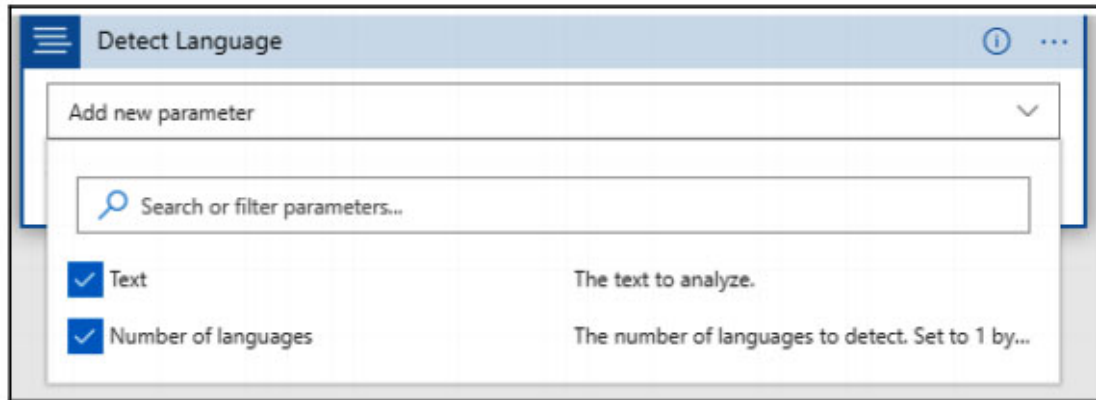
*Figure 4.24: Select OCR*

## Search for Text Analytics



*Figure 4.25: Search Text Analytics*

**Step 9:** In the **Detect Language** window, add a new parameter.



*Figure 4.26: Add a new parameter*

Repeat the same for detecting the Key Phrases and detecting Sentiment.

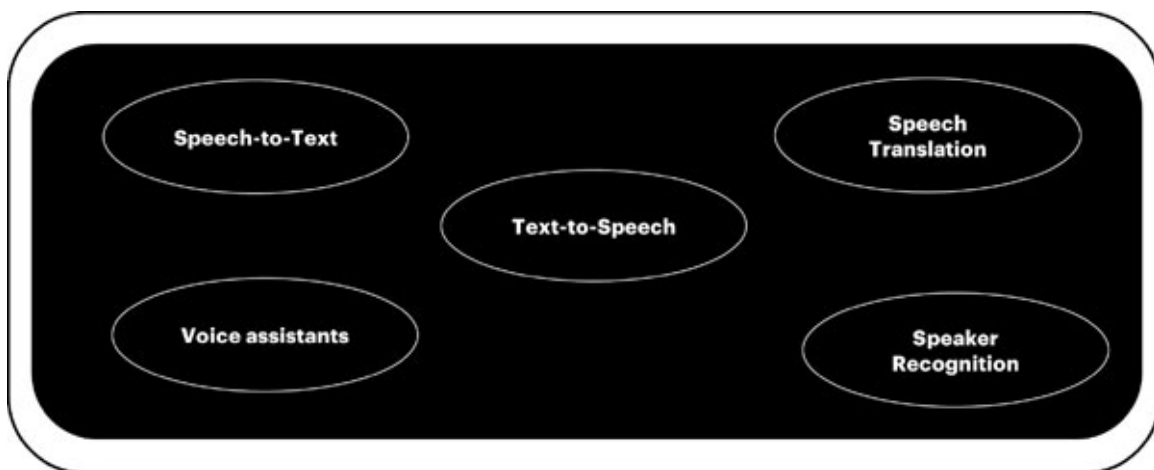
**Step 10:** Now configure a step in the logic app to save the results in a file in a drop box folder.

**Step 11:** Create an Azure Functions App that will pick from Drop Box and update in Azure SQL Database.

**Step 12:** Create a Power BI dashboard to show reports. Create a custom portal to show the results.

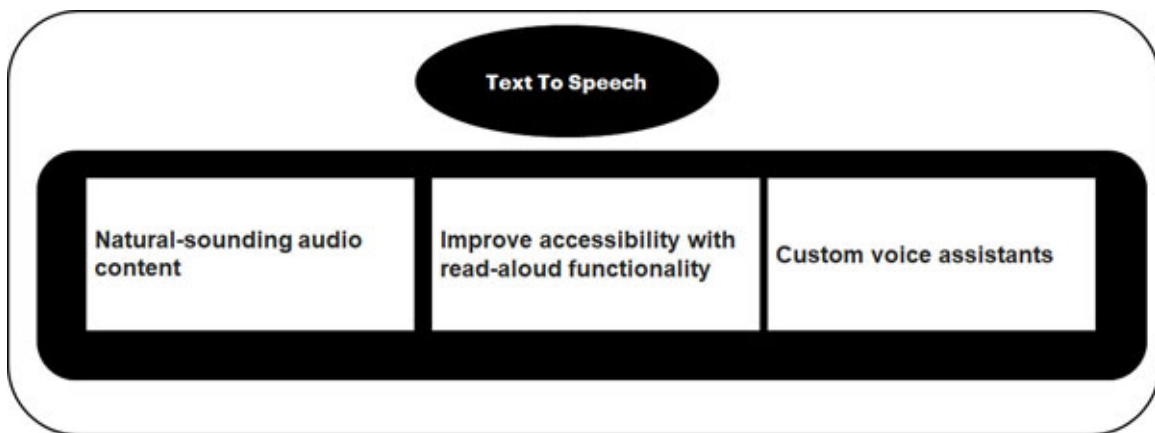
## Speech

The speech services are used to detect and analyze voice-based content from the users. These services can be used for various functionalities like converting text to speech, speech to text transcriptions, recognizing speaker, adding a voice to the application etc. as depicted as follows:



*Figure 4.27: Speech Services*

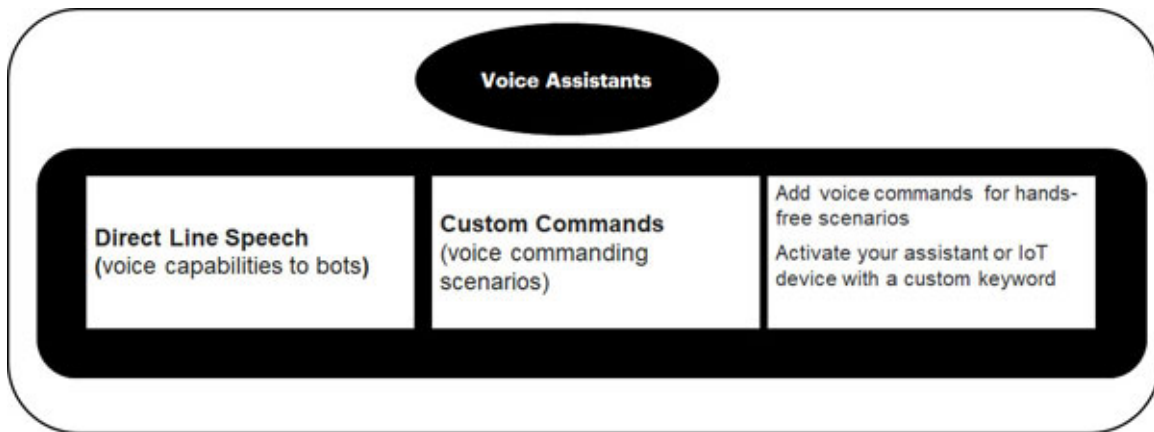
- **Speech to Text:** Speech to Text helps gain customer insights with call center transcription, improve experiences with voice-enabled assistants, capture key discussions in meetings and more, translate conversation to text. NLP can also be used to make the interpretations more intelligent.
- **Text to Speech:** Text to Speech service helps convert text which is a natural human live voice with automated modulation. This helps in creating apps and services that can have conversations. Text To Speech supports 270 neural voices across 119 languages. Some of the use cases are given below:



*Figure 4.28: TextToSpeech*

- **Speech Translation:** This service helps translate in real time, as users are speaking, speeches from one language to another. It supports more than 30 languages.
- **Speaker Recognition:** This is still in preview mode. It helps identify speeches from the person and identify the speakers based on the learning.
- **Voice Assistants:** Voice Assistants provide fast and reliable interaction between an assistant and a device.

Some of the use cases for voice assistants are given as follows:



*Figure 4.29: Voice Assistants*

The use cases for voice assistants include creating a voice assistant for an app, creating a touchless, voice-first experience to improve safety and support back-to-work scenarios, activating with a custom key word etc.

## **Bing Web Search**

The Bing web search APIs can be used to build connected apps and services that can integrate search capabilities like News Search, Web Search, Image Search. We can build applications that can integrate images, web pages, location, news without advertisements.

These APIs can also help in auto suggesting search terms, filtering results by content type, highlighting, localizing search results by market, region and or country and analyzing search metrics with Bing Statistics.

## **Lab**

**Step 1:** Go to Azure Portal. Search for Bing Search. It brings up the following Window:

[Home](#) > [Create a resource](#) >

# Bing Search v7 ...

Microsoft



## Bing Search v7 [Add to Favorites](#)


Microsoft


★ 4.0 (2 Azure ratings)

Create

*Figure 4.30: Azure Portal-Bing Search*

**Step 2:** The deployment success as given below comes up once the resource is deployed:

 Your deployment is complete



Deployment name: Microsoft.BingSearch

Subscription: [Visual Studio Professional Subscrip...](#)

Resource group: [mlstudio2](#)

Start time: 1/22/2022, 2:37:07 PM

Correlation ID: b7c8361f-e590-4893-beda-13d33e12...

▼

Deployment details [\(Download\)](#)

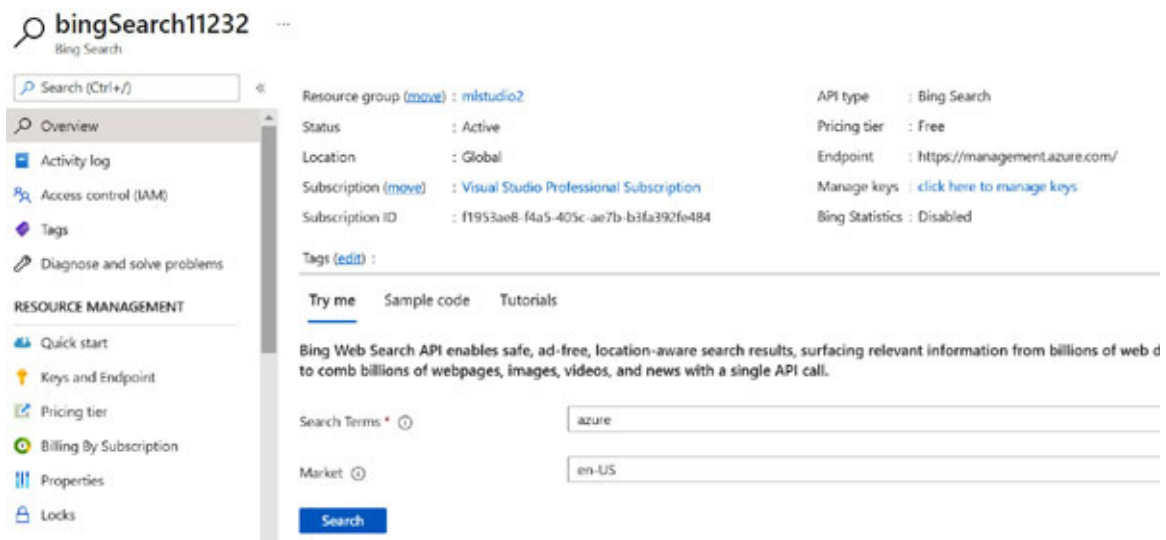
^

Next steps

[Go to resource](#)

*Figure 4.31: Deployment Window*

**Step 3:** Click on 'Go To Resource' button and add a Search Term and click on Search:



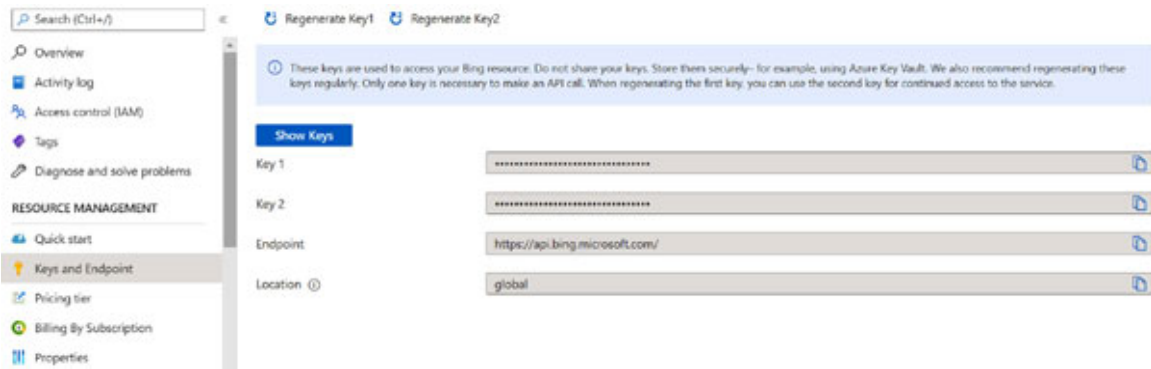
*Figure 4.32: Add a Search Term*

**Step 4:** The Search results appear as follows:



*Figure 4.33: Search Results Window*

**Step 5:** For programmatic invocation we can get the Key and use it in the code. The Key can be obtained as shown below:



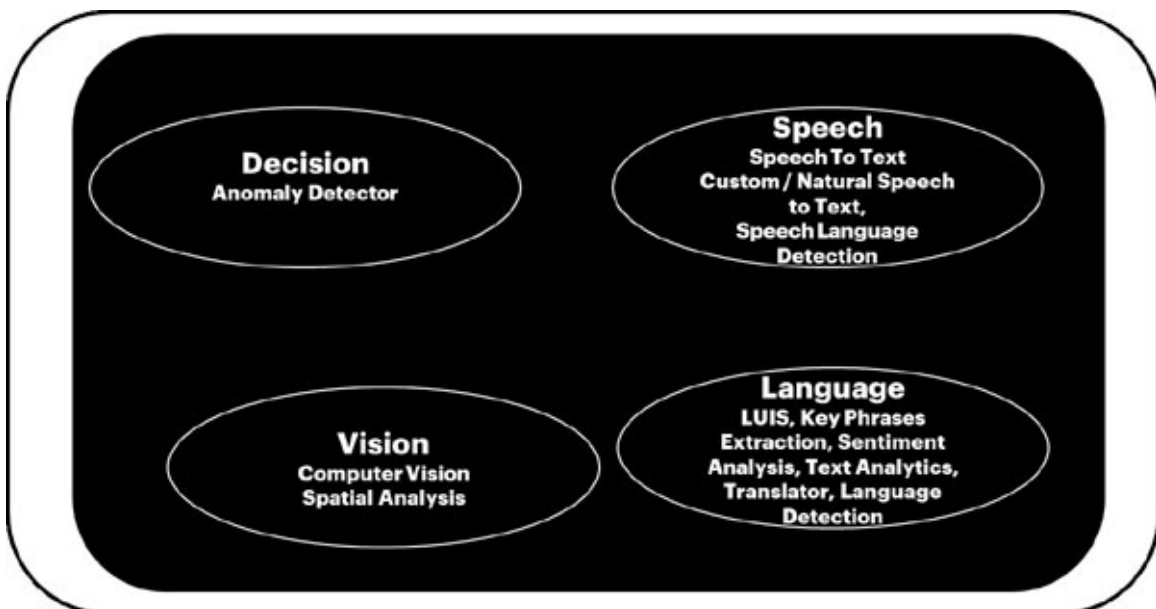
*Figure 4.34: Getting the key*

Now we will look at Docker Containers.

## **Azure Cognitive Services containers**

In addition to the cloud version of APIs and services Azure also provides Docker containers that let us use the same APIs on premise. These can be leveraged in scenarios where data compliance, security requirements etc hinder the usage of cloud APIs. This helps in creation of a portable application architecture. This can be deployed on-premises, on Azure and the edge.

Given below is a set of available Docker containers.



*Figure 4.35: Available Docker Containers*

## **Conclusion**



In this chapter, we have looked at the basics of using Decision, Language, Speech and Web Search Services of Azure Cognitive Services and Docker Containers. We have also learned about Azure Logic Apps and Azure Functions. In the next chapter, we will learn about Azure Applied Services.

## **Multiple Choice Questions**

Thought Experiment - This will be a set of questions around 7-10 to test the knowledge of the concepts learned in the chapter.

## **Questions**

- a. **Which service is used to make good decisions using Decision APIs to analyze the data and to derive insights from the same.**
  - a. Decision
  - b. Analyse
  - c. Detect
  - d. Insights
- b. **Which service can be used to integrate search capabilities like Web Search, News Search, Image Search into custom applications?**
  - a. Decision
  - b. Analyze
  - c. Web Search
  - d. Image Search
- c. **Which service can be used to identify speeches from the person and identify them accordingly?**
  - a. Speaker Recognition
  - b. Speech To Text
  - c. Text To Speech
  - d. Voice Assistant
- d. **Which service is used to Gain customer insights with call center transcription, improve experiences with voice-enabled assistants, capture key discussions in meetings and more.**

- a. Speaker Recognition
  - b. Speech To Text
  - c. Text To Speech
  - d. Voice Assistant
- e. Which service is used to Give app a voice**
- a. Speaker Recognition
  - b. Speech To Text
  - c. Text To Speech
  - d. Voice Assistant
- f. Which service can be used to translate audio from more than 30 languages ?**
- a. Speech Translation
  - b. Speech to Language
  - c. Voice Assistant
  - d. Text To Speech
- g. Which service can be used to create apps and services that speak conversationally ?**
- a. Speech Translation
  - b. Speech to Language
  - c. Voice Assistant
  - d. Speech To Text
- h. Which service Azure Content Moderator cognitive service verifies and flags text, image, and video content for material that is offensive with labels?**
- a. Azure Content Sanitizer
  - b. Analyze
  - c. Web Search
  - d. Azure Content Moderator
- i. Which API can be used to extract a list of strings denoting the key talking points in the input text. ?**

- a. Decision
  - b. Analyze
  - c. Key Phrase Extraction
  - d. Language Detection
- j. Which service can be used to text document as input and returns language identifiers
- a. Decision
  - b. Analyze
  - c. Key Phrase Extraction
  - d. Language Detection

## **Answers**

You have images of the movie and images of the audience coming out of the theatre.

1. **a. Decision**
2. **b. Web Search**
3. **a. Speaker Recognition**
4. **b. Speech To Text**
5. **d. Voice Assistant**
6. **d. Text To Speech**
7. **c. Voice Assistant**
8. **b. Video Indexer Insights**
9. **c. Key Phrase Extraction**
10. **d. Language Detection**

# CHAPTER 5

## Azure Applied AI Services

After reading this chapter, the reader will get an overview of Azure Applied AI Services which include Azure Form Recognizer, Azure Metrics Advisor, Azure Cognitive Search, Azure Immersive Reader, Azure Video Analyzer for Media and Azure Bot Service. Azure Applied AI Services help developers apply AI to solve business problems very easily. They are built on top of Azure Cognitive Services. They optimize tasks like anomaly detection, monitoring etc.

### Structure

The following topics will be covered in this chapter:

- Overview of Azure Applied AI Services
- Azure Metrics Advisor
- Azure Immersive Reader
- Azure Video Analyzer for Media
- Azure Form Recognizer
- Azure Cognitive Search

### Objective

By the end of this chapter, the reader will understand Azure Applied AI Services and how to effectively leverage these services.

### Overview of Azure Applied AI Services

Following is an overview of Azure Applied AI Services:

- **Azure Metrics Advisor:** It helps in identifying and fixing problems leveraging a combination of near-real-time monitoring and adapting

models to specific scenarios thereby offering granular analysis with alerting and diagnostics.

It uses AI to perform data monitoring and anomaly detection in time series data. The service automates the process of applying models to data, and provides a set of APIs and a web-based workspace for anomaly detection, data ingestion, and diagnostics. Users can build AIOps, predictive maintenance, and business monitor applications on top of the service.

- **Azure Immersive Reader:** It is an inclusively designed tool that implements techniques to improve reading comprehension for language learners, new readers, and people with learning differences.
- **Azure Form Recognizer:** It helps in identifying and extracting text, key/value pairs, and table data from form documents. It helps in ingesting text from forms and outputs structured data that includes the relationships in the original file.
  - **Azure Video Analyzer for Media:** It helps extract metadata such as spoken words, faces etc. from video and audio files.
  - **Azure Cognitive Search:** It leverages built-in AI capabilities that help in identifying and exploring relevant content.
  - **Azure Bot Service:** It helps develop conversational AI experiences.

We will look at each of the services in detail.

## [Azure Metrics Advisor](#)

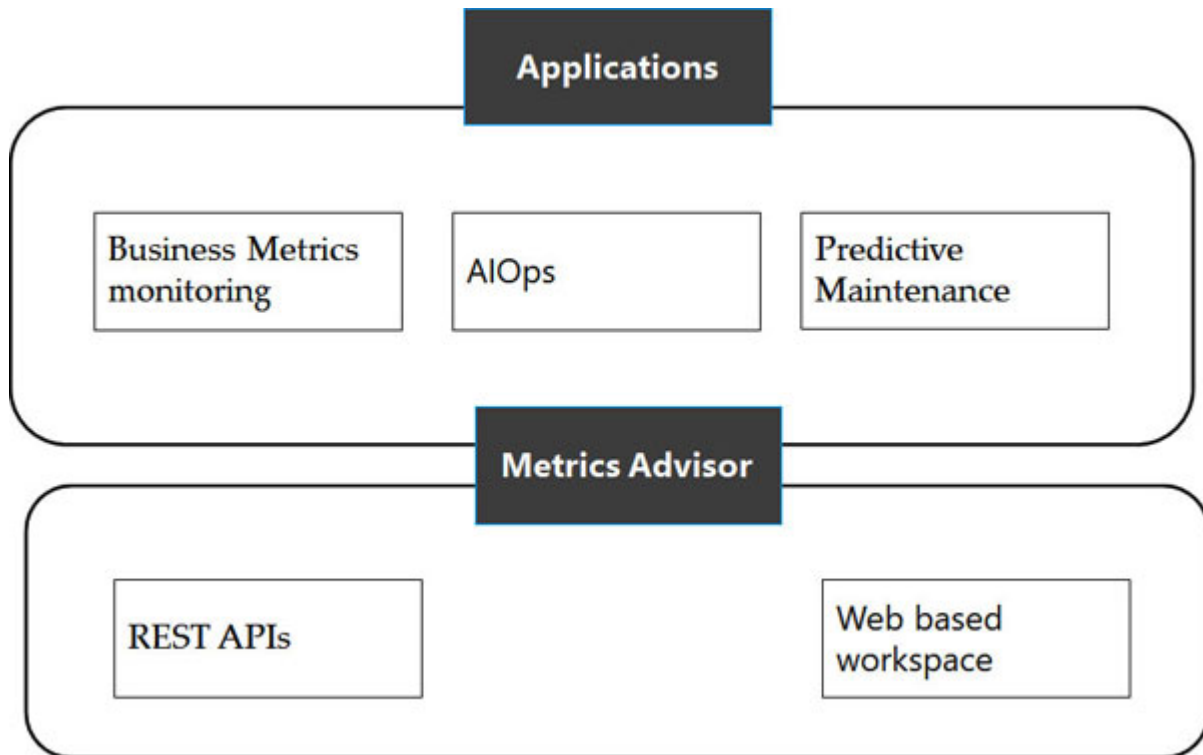
Azure Metrics Advisor helps analyze time-series real time data and surface anomalies, and helps derive insights and create alerts. The Azure Metrics Advisor automatically applies models on the data. It also provides a set of web based workspace and APIs for anomaly detection, data ingestion and diagnostics.

Azure Metrics Advisor helps developers with capabilities of multi-dimensional metric data ingestion, automatic model customization and anomaly detection. The capabilities of the pipeline can be easily used by developers to build artificial intelligence for IT operations, predictive maintenance, and business metric monitoring solutions.

Azure Metrics Advisor helps in the following:

- Analyze multi-dimensional data from multiple data sources
- Identify and correlate anomalies
- Configure and fine-tune anomaly detection model used on your data
- Diagnose anomalies and help with root cause analysis
  - Best model for the data is selected by Metrics Advisor
  - Multi-dimensional metrics can be used to monitor time series
  - Model can be customized using parameter tuning and interactive feedback
- Notification through multiple channels using hooks like email hooks, Azure Devops Hooks, Teams hooks, etc.
- **Analyze Root Cause:** Metrics Advisor can combine anomalies detected into a diagnostic tree on the same multi-dimensional metric to help analyze root cause.
- Smart diagnostic insights: Smart diagnostic insights enable to help catch abnormal status using all metrics which are related

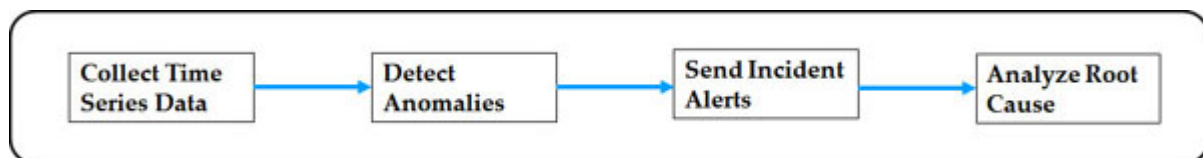
Following is a view on how Azure Metrics Advisor can be leveraged:



*Figure 5.1: Azure Metrics Advisor use cases*

Azure Metrics Advisor leverages machine learning to detect anomalies, predict future anomalies, and help in development of systems that detect and act on anomalies. It can ingest data from a number of data sources, leverage the data to build models, tune the models, and use feedback to customize the models. It also has the capability to perform root cause analysis with insights and recommend appropriate actions.

Following is the flow diagram for Azure Metrics Advisor:



*Figure 5.2: Azure Metrics Advisor Flow Diagram*

Following are a set of recommendations that are commonly used:

- a. **High availability recommendations:** Help ensure the continuity of critical applications. High availability recommendations can help in the following areas:

- Identify virtual machines not part of an availability set and suggest/recommend moving them to an availability set.
- Identify application gateway instances that are not configured for fault tolerance to ensure business continuity of critical applications.
- Prevent accidental deletion of virtual machine data.
- Improve the performance of virtual machine disks.

#### b. **Security recommendations**

Advisor integrates with Azure Security Center. It provides **security recommendations**, helping detect, respond to and prevent security threats.

#### c. **Cost recommendations**

Helps in reducing virtual machine costs and identifying cost-effective solutions for managing multiple SQL databases.

#### d. **Performance recommendations** help improve performance of applications. Performance recommendations can help in the following areas:

- Improve database performance
- Improve Redis Cache performance
- Improve App Service performance

Given below are the steps for creating a workflow for Azure Metrics Advisor:

- Create an Azure resource for Metrics Advisor
- Build monitor using web portal
- Onboard data
- Change configuration for anomaly detection to address the specific needs
- Subscribe anomalies for notification
- Diagnostic insights can be viewed
- REST API can be used to customize instances

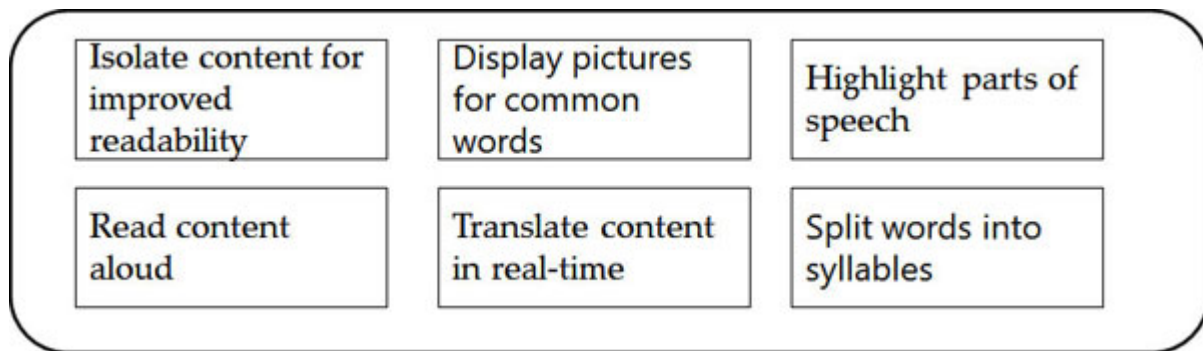


## Azure Immersive Reader

Azure Immersive Reader helps in improving comprehension for language learners, new readers, and people with learning differences such as dyslexia. It helps in making reading easier. Immersive Reader client library can be leveraged to improve web applications.

Immersive Reader is a standalone web application which can be integrated in a web application in an iframe by invoking the Immersive Reader client library. The Immersive Reader client library handles the iframe and interaction with the Immersive Reader backend service.

Some of the features of the Azure Immersive Reader are listed as follows:



*Figure 5.3: Azure Immersive Reader Features*

1. **Isolate Content:** This helps in improving readability
2. **Display Pictures:** It can display pictures for commonly occurring words
3. **Highlight Parts of Speech:** It can highlight parts of speech
4. **Read Content Aloud:** It allows users to select text to read out loud
5. **Translate Content in real time:** It helps in real time translation of text into many languages
6. **Split words into Syllables:** It can help in breaking words into syllables

## Azure Video Analyzer for Media

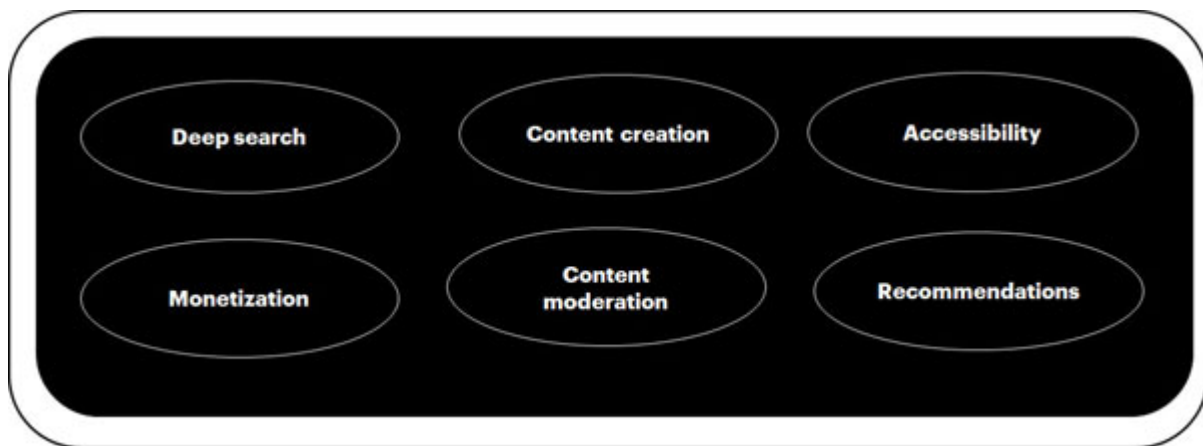
Video Analyzer for Media helps users with getting detailed insights on videos which include summarized insights that provide an aggregated view of faces, emotions, and topics. These insights can be used for content

creation such as trailers, enhancing search across a video library, providing relevant ads based on extracted insights, content moderation, and recommendations.

It also helps in extracting insights from videos and enhancing content delivery experiences by detecting faces, spoken words, emotions and characters. Recommendations can be added and clips can be highlighted thus improving the engagement levels. Furthermore, workflows can be triggered when the Video Analyser sees, detects or hears what the user is looking for.

This helps with face detection, celebrity identification, custom face recognition, thumbnail extraction, visual text recognition, object identification, visual content moderation, shot detection, black frame detection, key frame detection, audio analysis, keyword extraction, topic inference, sentiment analysis, emotion detection, speech to text conversion, language detection, noise reduction, speaker identification, and speaker statistics.

Some of the scenarios where insights provided by Video Analyser are given as follows:

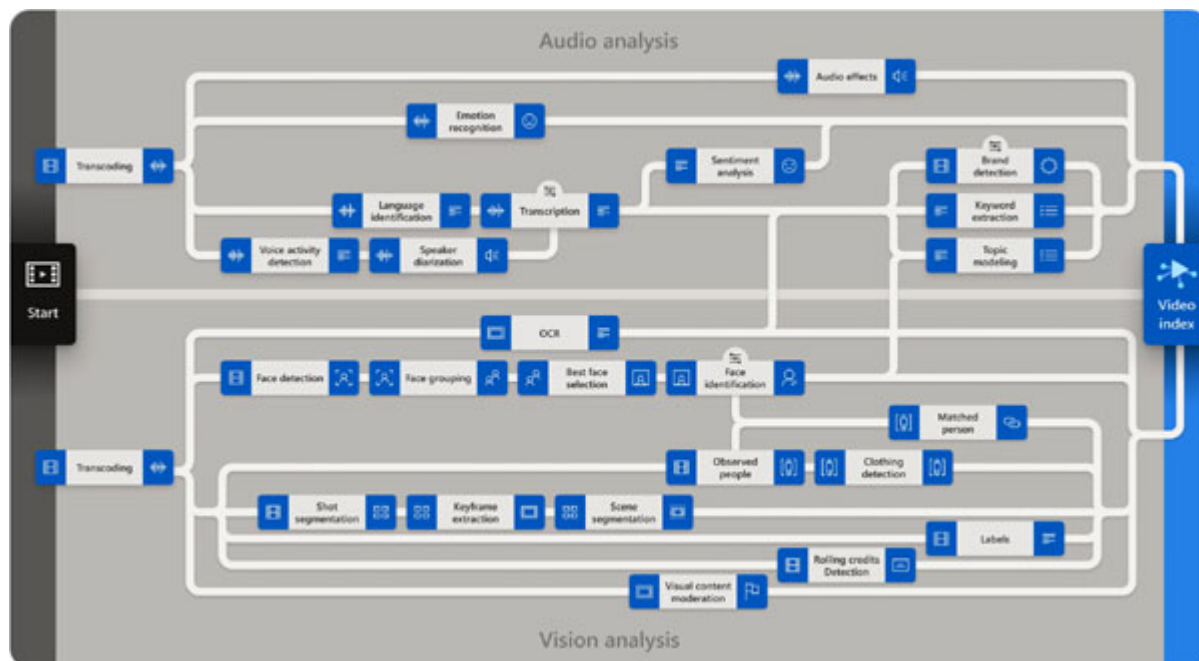


*Figure 5.4: Azure Video Analyzer for Media*

- **Deep search:** Insights provided by Azure Video Analyser can help enhance the search experience.
- **Content creation:** We can use social media content, highlight reels, or news clips and create trailers based on the insights Video Analyzer for Media provides.

- **Accessibility:** The transcription and translation provided by Video Analyzer for Media can be used to enable accessibility to users.
- **Monetization:** By delivering relevant ads, Video Analyzer for Media can be leveraged to help increase the value of videos and create revenue opportunities.
- **Content moderation:** We can use content moderation to alert users or block objectionable content/videos thus making the content safe.
- **Recommendations:** By highlighting the relevant video moments to users video insights can be used to improve user engagement.

The diagram below gives a view on how Azure Media Analyser works:



*Figure 5.5: Azure Immersive Reader*

Image from: <https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/video-indexer-overview>

- Video / Audio Insights

The following table describes in detail the features provided by Video / Audio Indexer.

Feature	Description
Face detection	Helps detect faces and group faces in the video

Celebrity Detection	Helps identifying celebrities in the video
Account-based face identification	Helps identify faces based on a trained model
Thumbnail extraction for faces	Helps identify the best face in a group of faces captured
Visual text recognition	Helps identify text in videos using OCR
Visual content moderation	Helps moderate content
Labels identification	Helps identify objects and actions in a video
Scene segmentation	Helps detect scene change based on visual cues in a video
Shot detection	Helps detect shot change based on visual cues in a video
Black frame detection	Helps identify black frames in a video
Key frame extraction	Helps detect key frames in a video
Rolling credits	Helps detect rolling credits in videos
Animated characters detection	Helps detect, group and recognize characters in animated content
Editorial shot type detection	Helps tag shots like close up etc.
Observed People Tracking (preview)	Helps detect observed people in the video and provides information like location, timestamp etc. of the observed person
Matched person	Help match between detected faces and people observed in the video

**Table 5.1: Video / Audio Insights**

- **Audio Insights**

The following table describes in detail the features provided by Audio Insights.

Feature	Description
Audio transcription	Helps Convert speech to text
Automatic language detection	Helps detect the dominant language
Multi-language speech identification and transcription	Helps identify the spoken language in different segments from audio
Closed captioning	Helps create captions
Two channel processing	Helps merging transcripts

Noise reduction	Helps clear up noise
Transcript customization	Helps train custom speech to text models
Speaker enumeration	Helps understand which speaker spoke which words and when
Speaker statistics	Helps provide speech ratio statistics
Textual content moderation	Helps detect text in the audio transcript
Audio effects	Helps identify audio effects like speech, silence, claps
Emotion detection	Helps identify emotions
Translation	Helps create translations of the audio transcripts
Audio effects detection (preview)	Helps detect audio effects like laughter

*Table 5.2: Audio Insights*

- **Video and Audio Insights**

The following features are available in Video and Audio Insights:

Feature	Description
Keywords extraction	Helps extract keywords from speech and visual text.
Named entities extraction	Leveraging NLP helps extract brands, locations, and people from speech and visual text.
Topic inference	Helps make inference of main topics from transcripts
Artifacts	Helps extract a rich set of “next level of details”.
Sentiment analysis	Helps identify neutral, positive and negative sentiments.

*Table 5.3: Video and Audio Insights*

## [Azure Form Recognizer](#)

Form Recognizer helps extract text and structure from documents. It helps in extracting text, key/value pairs, and tables from documents, forms, and receipts using pre-built and unsupervised building learning. relationships between fields and entries in the documents.

Following models are supported by the Form Recognizer:

- **General document:** Helps in analysing and extracting tables, text, key value pairs, structure, and named entities.

- **Layout:** Helps in Analyzing and extracting tables, words, lines and selection marks in documents and forms documents.
- **Custom:** Helps in analyzing and extracting form fields and other content from custom forms using custom trained models.
- **Invoices:** Helps in analyzing and extracting common fields from invoices, using a pre-trained model for invoice.
- **Receipts:** Helps in analyzing and extracting common fields from receipts, using a pre-trained model for receipt.
- **ID documents:** Helps in analyzing and extracting common fields from ID documents like passports or driver's licenses, using a pre-trained model for documents.
- **Business Cards:** Helps in analyzing and extracting common fields from business cards, using a pre-trained model for business cards.

### **Prebuilt Models:**

Analyze and extract data from common document types, using a pre-trained model.

- **Pre Built-businessCard:** Helps in extracting text and key information from business cards.
- **Prebuilt-idDocument:** Helps in extractintext and key information from driver licenses and international passports.
- **Prebuilt-receipt:** Helps in extracting text and key information from receipts.
- **Custom Models**

In addition to the support for pre-built models, we can train a custom model using sample input forms. The minimum number of forms required for training is 5. The model uses unsupervised learning to identify fields, tables, entries and the relationships in the form.

### **Accessing Form Recognizer**

Form Recognizer is accessible using REST APIs. Form Recognizer is currently in limited access preview.

In this short lab, we see how we can use Form Recognizer studio to analyze a sample invoice.

We will use The Layout API in this lab. The Layout API analyzes and extracts text, tables and headers, selection marks, and structures information from forms and documents.

## Configure service resource

To use Form Recognizer, you need an Azure subscription containing a service resource for usage and billing. Resources are organized in resource groups.

[Learn more](#)

Subscription \*

Resource group \*

[Create new](#)

Form Recognizer or Cognitive Service Resource \*

☒ Create new resource

Name \*

[Continue](#)

*Figure 5.6: Configure Service Resource*

Following is a sample invoice document:



Analyze

API version: 2021-09-30-preview

☕

**Liberty's Delightful Sinful Bakery & Café**  
765 Halifax St. Clearwater, FL 33756

Our reference: 3456623      Your reference: 2334566

**Received from:**  
8556 Indian Summer Ave.  
New Haven, CT 06511

**Liberty booking contact**  
40 River Street  
East Northport, NY 11731

Name Summer River  
Tel. 34456632  
E-mail [email6@libertydelightfulsinful.com](mailto:email6@libertydelightfulsinful.com)

### Booking Confirmation – ORIGINAL

Our reference:	<b>3456623</b>	Booking date:	05-Dec-2018
Your reference:	2334566	Contract No:	334566
BL/NO:	EURH234		
Summary:	45x72	<input type="checkbox"/> Opt. A	<input type="checkbox"/> Opt. B

Figure 5.7: Sample invoice

Following is the sample output displayed:

```
{
  "apiVersion": "2021-09-30-preview",
  "modelId": "prebuilt-layout",
  "stringIndexType": "textElements",
  "content": "Liberty's Delightful Sinful Bakery & Cafe\n765 Halifax St. Clearwater, FL 33756\nOur reference: 3456623\nYour reference: 2334566\nReceived from:\n8556 Indian Summer Ave.\nNew Haven, CT 06511\nName Summer River\nTel. 34456632\nE-mail email6@libertydelightfulsinful.com\nbooking Confirmation - ORIGINAL\nOur reference:\nYour reference:\nBL/NO:\n3456623\n2334566\nEURH234\n45x72\nLiberty booking contact\n40 River Street\nEast Northport, NY 11731\nBooking date: 05-Dec-2018\nContract No:\n334566\nSummary:\nExport:\nCommon\nExport empty pick up depot (s)\n97 Morris Lane\nSterling Heights, MI 48310\nOpt. A\nOpt. B\nFrom\nTo\nBy\nETD\nETA\n118 Queen Street\nHoboken, NJ 07030\n52 West Trenton St.\nHarleysville, PA\n19438\ncause science slow\n09-Dec-2018\n19:00\n09-Dec-2020\n11:00\n9 Ketch Harbour\nAve.\nVincetown, NJ\n75 Fawn Street\nPeabody, MA 01960\ntone late spoken\n12-Dec-2018\n10:00\n19-Dec-2020\nDeadline\nLocation\nDate/Time (local)\nRequired action\nTable\nHarleysville\n(PA)\n08-Dec-2019\nNobody loves a pig\nwearing lipstick.\nFlight\nHarleysville\n(PA)\n08-Dec-2019\n13:00\nTwo more days and all his\nproblems would be solved.\nRound\nHarleysville\n(PA)\n09-Dec-2019\nAccent\nHarleysville\n(PA)\n10-Dec-2019\nMonkey\nHarleysville\n(PA)\n11-Dec-2019\nRoute\nHarleysville\n(PA)\n11-Dec-2019\nPeanuts don't grow on \ntrees.\n\nunselected:\n\nunselected:"
```

Figure 5.8: Output Generated

## Azure Cognitive Search

Azure Cognitive Search is a search service in the cloud that gives a rich search experience. It can pull data from multiple sources into web, mobile

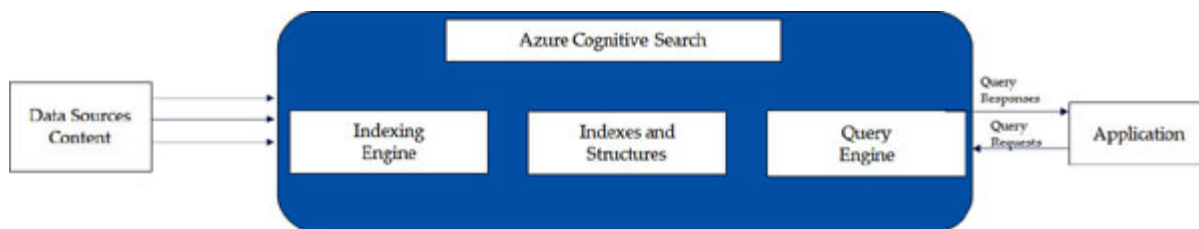


and other applications. It can search data using AI and natural language processing. It has the capability for data enrichment and encryption. Features include autocomplete, filters, paging, facets and boosting. It helps in easy implementation of faceted navigation, autocomplete, relevance tuning, filters.

It can help in searching Azure Blob Storage, Cosmos DB.

Azure Cognitive Search supports Microsoft's natural language processors and Lucene analyzers and. We can configure analyzers to achieve specialized processing of raw content, such as filtering out diacritics, or recognizing and preserving patterns in strings.

Following is the architecture of Azure Cognitive Search:



*Figure 5.9: Azure Cognitive Search*

A search service helps in indexing and searching unindexed data when the client app sends query requests and sends the response.

Components provided by Search Service include:

**Data Sources/Content supported:** The data sources supported include SQL Server on Azure VM, Azure SQL Database, Cosmos DB, Managed Instance, Azure Table Storage, Azure blob storage container, Azure Data Lake Storage Gen2, or SharePoint Online (preview), or any dataset that contains JSON documents.

**Index:** Azure Cognitive Search creates the index on the specified data. The Indexer runs at scheduled intervals.

**Querying:** Client applications can send and receive a response from the search service Once the index is populated with search text.

The search service includes synonym matching, auto-complete, filter, fuzzy matching, auto spell correction, sort, or pattern matching, identification of visual features, such as facial detection, **Optical Character Recognition (OCR)**, and image recognition and image interpretation.

We can create a basic search service using either of the two approaches:

### **Approach 1:**

- Decide on a tier. For advanced capabilities, we will need a billable tier else we can go with a free tier.
- In the Azure portal create a search service.
- Import data using Import data wizard. Give data source details as input and create, load, and query an index.
- Use Search Explorer portal client to query the created search index.

### **Approach 2:**

- Create a search index using the NET SDK, REST API, .portal, or another SDK. The index schema defines the structure of searchable content.
- Upload content using the “push” model to push JSON documents from any source, or use the “pull” model (indexers) if your source data is of a supported type.
- Use Search Explorer portal client to query an index or use REST API, .NET SDK, or another SDK to query an index.

## **Lab: Creating a Azure Cognitive Search Service**

In this sample, we will leverage Approach 1 where we will create a search service using the portal, import data from a sample database, create an index and use Search Explorer to query the index.

### **Prerequisites:**

An Azure account with an active subscription.

Following are the steps to be followed for the lab in detail:

**Step 1 - Create a Search Service:** Sign in to the Azure portal with your Azure account.

## Create a search service

### Basics

Subscription	Visual Studio Professional Subscription
Resource Group	rg-az-12
Location	West US 2
Service name	(new) searchservice2
Pricing tier	standard (25 GB/Partition*, max 12 replicas, max 12 partitions, max 36 search units)
Estimated cost per month	\$12,000.00

### Scale

Replicas	1
Partitions	1

### Networking

Endpoint connectivity (data)	Public
------------------------------	--------

*Figure 5.10: Creating Search Service*

## Step 2: Start the Import data wizard and choose a data source

In this lab, we are choosing a sample database.

Screenshot of the Import data command is given below: It shows the available data sources.

**Connect to your data** Add cognitive skills (Optional) Customize target index Create an indexer

Create and load a search index using data from an external data source. Azure Cognitive Search crawls the data structure you provide, it into an index. [Learn more](#)

Data Source

Name ↑↓

No results.

Existing data source

Existing data source

Samples

Azure SQL Database

SQL Server on Azure VMs

Azure Cosmos DB

Azure Blob Storage

Azure Data Lake Storage Gen2

Azure Table Storage

SharePoint Online (preview)

Next: Add cognitive skills (Optional)



**Figure 5.11: Data Sources Available**

Choose sample database as shown below:

**Connect to your data**   Add cognitive skills (Optional)   Customize target index   Create an indexer

Create and load a search index using data from an external data source. Azure Cognitive Search crawls the data structure you provide, extracts it into an index. [Learn more](#)

Data Source: Samples

Type	Name
	realestate-us-sample
	hotels-sample

**Figure 5.12: Select Sample Data Source for this lab**

**Creating Index:** Index has to be created before loading data. An index requires one of the fields that should be assigned as the document key to identify each document uniquely. We can specify language analyzers and suggesters in addition to help in suggested and autocomplete queries.

Given below is Create Index screenshot:


**Add index** ...

Index name \* ⓘ indexer1 ✓

Key \* ⓘ id ▼

Suggester name  Search mode ⓘ analyzingInfixMatching ▼

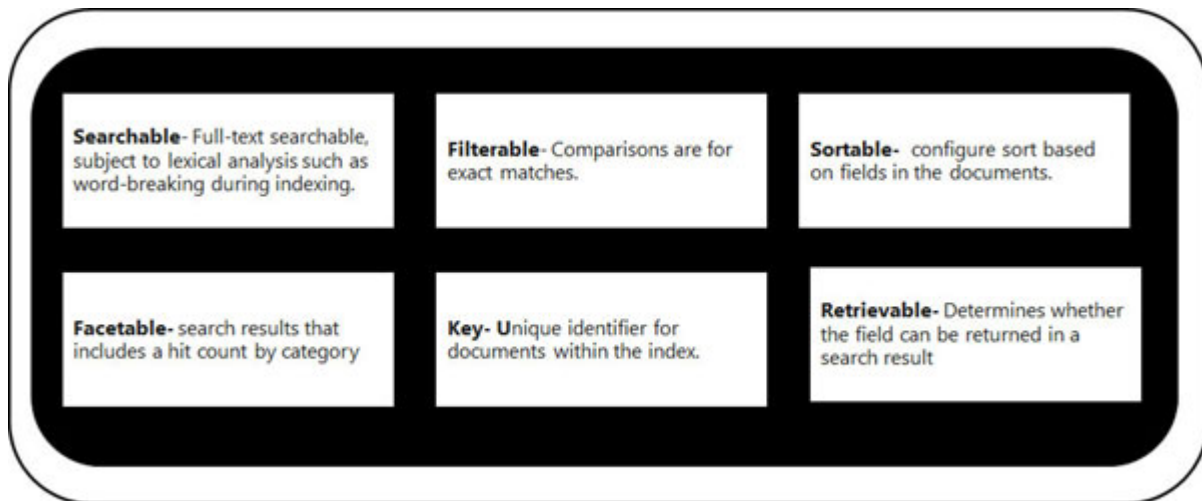
+ Add field   + Add subfield   Delete

Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable	Analyzer	Suggester
 id	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/> ...

**Figure 5.13: Add Index**

Index attributes aid in controlling how the field is used.

Following are the ways index attributes can be used:



*Figure 5.14: Index attributes*

- Searchable determines that a field is included in full text search.
- Filterable, Facetable, and Sortable determine if fields are used in a filter, faceted navigation or sort structure.
- Key is a unique document identifier.
- Retrievable makes the search result return the field.

**Cognitive skills:** We can add cognitive skills to the Search Service as shown below:

Connect to your data   Add cognitive skills (Optional)   Customize target index   Create an indexer

⚠ Enrich and extract structure from your documents through cognitive skills using the same AI algorithms that power Cognitive Search. Optionally, save enriched documents in Azure storage for use in scenarios other than search. [Learn more](#)

---

✓ Attach Cognitive Services

---

✓ Add enrichments

---

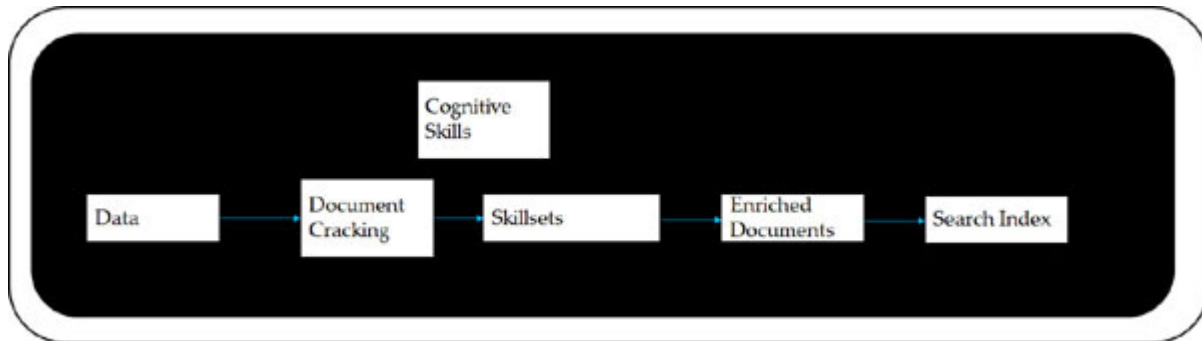
✓ Save enrichments to a knowledge store

---

Previous: Connect to your data   Skip to: Customize target index

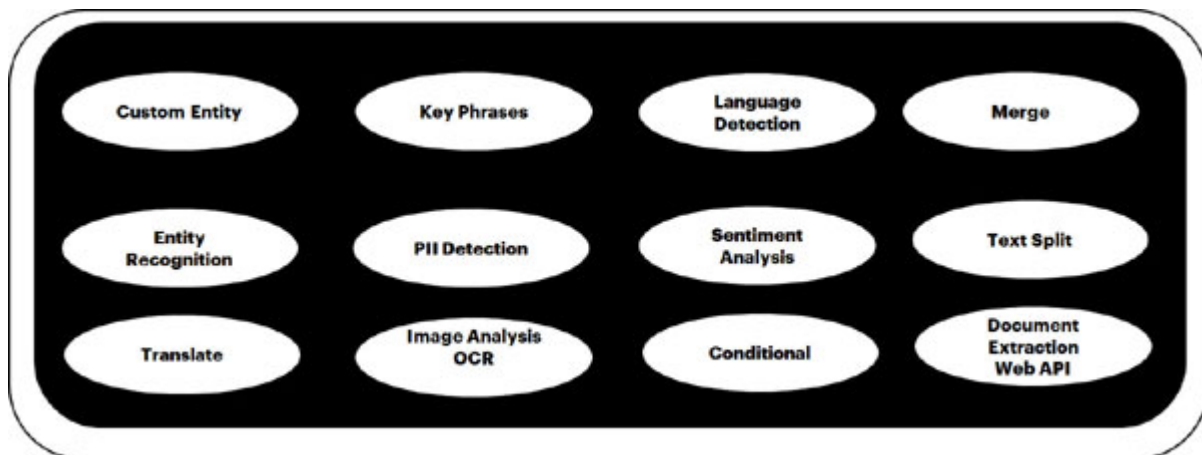
*Figure 5.15: Cognitive Skills*

**Add Enrichments:** A **skillset** can be attached to an indexer which does the AI processing. The indexer extracts the content and sets up the AI Processing pipeline. It analyzes, identifies, and creates new information out of content like images, blobs, and raw text. Output is a search index.



*Figure 5.16: Skillset*

**Attach Cognitive Skills:** Some of the cognitive skills provided by the Azure Cognitive search are described below:



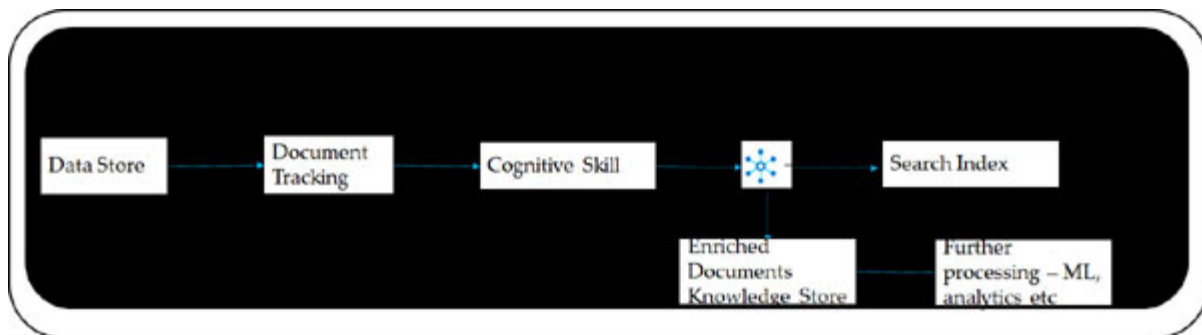
*Figure 5.17: Cognitive Skills*

- **Custom Entity:** Helps in looking for a user-defined set of words and phrases.
- **Key Phrases:** Helps in extracting data like important phrases using a pre-trained model. It is useful to retrieve the main points in a record.
- **Language Detection:** Helps in detecting language in each document.
- **Merge:** Helps in combining text from multiple fields to a single field. A common use case is to extract a caption from the OCR skill and

then merge it with the content field of a document.

- **Entity Recognition:** Helps in identifying location, people, emails, organization, etc.
- **PII Detection:** Helps in extracting personally identifiable information from a given text.
- **Sentiment Analysis:** Helps in retrieving score based on positive or negative or neutral sentiment.
- **Text Split:** Helps in enriching content by splitting the text into sentences.
- **Translate:** Helps in translating a text into a variety of languages.
- **Image Analysis:** Helps in Identifying content of image and generating text description, generating tags and identifying landmarks or celebrities.
- **OCR:** Helps search faxes, images and scanned documents.
- **Conditional:** Allows merging, filtering, or assigning a default value based on a condition, or setting a default value for a value that does not exist.
- **Document Extraction:** Helps in extracting content from a document.
- **Web API:** Helps in Extension of AI enrichment pipeline.
- **Knowledge Store:**

**Save Enrichments to a Knowledge Store:** Knowledge Store helps in storing enriched content. It uses blob and tables and containers in Azure Storage. The data thus stored can be used for further independent analysis and for downstream processing for scenarios like knowledge mining.



*Figure 5.18: Enrichments*

The knowledge Store window looks as follows:

## Import data ...

### Save enrichments to a knowledge store

A knowledge store allows you to project your enriched documents into tables and blobs. [Learn more about Knowledge Store](#)

#### Azure table projections

☐ Documents

#### Azure blob projections

☐ Document

#### Knowledge Store Power BI analytics report

Visualize the data from Knowledge Store with Power BI.



Figure 5.19: Knowledge Store

## Step 4 - Configure indexer

- In the Import data wizard, click **Indexer** | **Name**.
- Type an indexer name.
- Click **submit** to create the indexer.

Configure Indexer Window looks as follows:

[Connect to your data](#) [Add cognitive skills \(Optional\)](#) [\\*Customize target index](#) [Create an indexer](#)

**Indexer**

**Name \***  ✓

⚠ A schedule can't be configured on samples or existing data sources without change tracking. Edit your data source to add change tracking if you wish to set up a schedule.

**Schedule** ⓘ Once Hourly Daily Custom

**Description**

[Previous: Customize target index](#) [Submit](#)



*Figure 5.20: Configure Indexer*

## Monitor progress

We can monitor the progress by going to the Indexers list.

Go to the Overview page and click on the tab named **Indexers**.

Usage

Monitoring

Indexes

Indexers

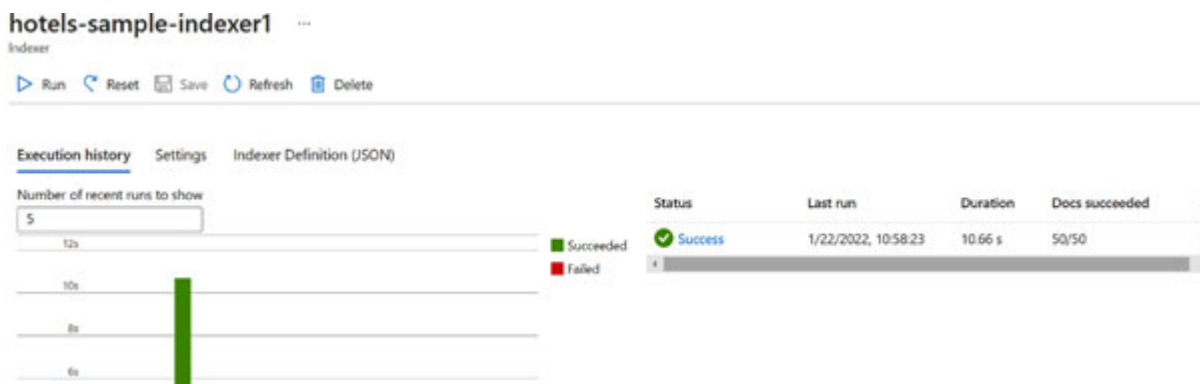
Data sources

Skillsets

Status	↑↓	Name	↑↓	Last run	↑↓	Docs succeeded
<div><div></div>In progress</div>		hotels-sample-indexer		Just now		0/0

*Figure 5.21: Monitor Progress*

Click on the indexer to see more details about the indexer's runs.



hotels-sample-indexer1 ...

Indexer

Run Reset Save Refresh Delete

Execution history Settings Indexer Definition (JSON)

Number of recent runs to show: 5

12s 10s 8s 6s

Execution history table:

Status	Last run	Duration	Docs succeeded
Success	1/22/2022, 10:58:23	10.66 s	50/50

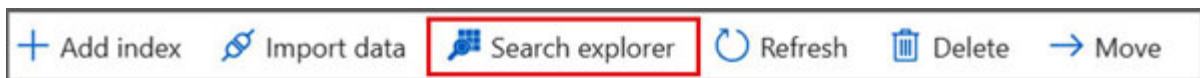
*Figure 5.22: Run Window for Indexer*

## Query using Search explorer

You now have a search index that can be queried using **Search explorer**.

Search explorer sends REST calls that conform to the Search Documents API. The tool supports simple query syntax and full Lucene query parser.

Select **Search explorer** on the command bar.



*Figure 5.23: Search Explorer*

From Index, choose the sample index:

Home > Microsoft.Search > searchservice45 >

# Search explorer ...

searchservice45

Index

API version

hotels-sample-i... ▾	2021-04-30-Pre... ▾
----------------------	---------------------

*Figure 5.24: Choose index*

In the search bar, add a query string and select **Search**.

Query string ⓘ  
Examples: \*, \$top=10, \$top=10&\$skip=10&search=\* Search

*Figure 5.25: Add a query string*

The results are displayed in the results window:

Home > Microsoft.Search > searchservice45 >

Search explorer ...

searchservice45

Index API version

hotels-sample-i... ▾ 2021-04-30-Pre... ▾

Query string ⓘ

Examples: \*, \$top=10, \$top=10&\$skip=10&search=\* Search

Request URL

https://searchservice45.search.windows.net/indexes/hotels-sample-index/docs?api-version=2021-04-30-Preview&search=\*

Results

```
1 {
2   "@odata.context": "https://searchservice45.search.windows.net/indexes('hotels-sample-index')/$metadata#docs(*)",
3   "value": [
4     {
5       "@search.score": 1,
6       "HotelId": "45",
7       "HotelName": "Arcadia Resort & Restaurant",
```

*Figure 5.26: Search Explorer Output Window*

## Conclusion

In this chapter, we have looked at some of the Azure Applied AI services and how to leverage them. In the next chapter we will understand Bots in detail and how to use them.

## **Multiple choice questions**

This section consists of a set of questions that are designed to test your knowledge of the concepts covered in this chapter.

**a. Which Service helps extract text and structure from documents.?**

- a. Azure Machine Learning Studio
- b. Form Extractor
- c. Form Recognizer
- d. OCR

**b. What service helps users with getting detailed insights on videos which include summarized insights that provide an aggregated view of faces, emotions, and topics.**

drag and drop without writing any code?

- a. Automated ML
- b. Video Analyzer for Media
- c. Audio Extractor
- d. Frame Extractor

**c. What service helps in identifying and fixing problems leveraging a combination of near-real-time monitoring, adapting models to specific scenarios thereby offering granular analysis with alerting and diagnostics?**

- a. Azure Monitor
- b. Azure Metrics Advisor
- c. Azure Video Adapter
- d. All

**d. What service helps in improving comprehension for language learners, new readers,, and people with learning differences such as dyslexia.**

- a. Azure Immersive Reader
- b. Azure Video Reader
- c. Azure Reader

- d. Azure Pencil
- e. **What helps in identifying and extracting relevant content.**
  - a. Automation
  - b. Azure Cognitive Search
  - c. Azure Cognitive Extractor
  - d. Automated ML
- f. **What helps in storing enriched content for further analysis?**
  - a. Data Store
  - b. Knowledge Store
  - c. Enrichments Store
  - d. Content Database
- g. **What helps in extracting data like important phrases using a pretrained model?**
  - a. Automation
  - b. Key Phrases
  - c. Extractor
  - d. Phrase Extractor
- h. **What helps in looking for a user-defined set of words and phrases.**
  - a. Automation
  - b. Key Phrases
  - c. Extractor
  - d. Entity Recognition
- i. **What helps in AI processing.**
  - a. Automation
  - b. Key Phrases
  - c. Extractor
  - d. Skillset

**j. What helps developers with capabilities of multi-dimensional metric data ingestion, automatic model customization and anomaly detection**

- a. Azure Metrics Advisor
- b. Azure Cognitive Search
- c. Azure Cognitive Extractor
- d. Automated ML

**Sensitivity:** Internal & Restricted

## **Answers**

1. **c. Form Recognizer**
2. **b. Video Analyzer for Media**
3. **c. Azure Video Adapter**
4. **a. Azure Immersive Reader**
5. **b. Azure Cognitive Search**
6. **b. Knowledge Store**
7. **b. Key Phrases**
8. **d. Entity Recognition**
9. **d. Skillset**
10. **a. Azure Metrics Advisor**

# **CHAPTER 6**

## **Bots - A Brief Introduction**

This chapter will take you through the Microsoft Bot Landscape including the Bot Framework, the Bot Framework Composer after a brief introduction of Bots and how they have evolved over the years. We will look at different options have to offer, best practices and scenarios, a simple example using the Bot Framework using QnA and NLP (LUIS) capabilities as well and also understand how to develop/debug Bots locally with the Bot Emulator. We will also touch on the Bot Framework Composer and its features and a quick example.

### **Structure**

This chapter will cover the following topics:

- How Machines Learn – A Brief Introduction
- Introducing Bots
- The Bot Ecosystem
  - The Bot Framework
  - The Bot Framework Composer
  - Cognitive Intelligence
  - Search
  - Storage
  - Information Sourcesw
- Building the Bot Framework Composer
- Lifecycle
- Reference architecture
- The Bot Framework Emulator
- Enterprise Scenarios
- Typical use cases

## Objective

After reading this chapter, the reader will be able to understand the concept of Bots and will be able to develop Bots. They will be able to understand the concepts of NLP, LUIS, QnA Maker, Bot Composer and Bot Emulator.

## How Machines Learn – A Brief Introduction

The pragmatic machine based “*Imitation game*” developed by *Alan Turing* is arguably the first attempt at demonstrating machines exhibiting intelligence. But then, it was *John McCarthy*, the American computer scientist and inventor who is considered to be the pioneer and credited with coining the term “*Artificial Intelligence*” in 1955. He set himself the objective to explore ways to make a machine that could reason like a human. While AI is expected to do a lot of what Humans can do in about 50 years from now, we really do not know where we are headed. But one thing is for sure, that AI has moved from being a science fiction to research and to mainstream computing today.

## Introducing Bots

Bots are simply put, intelligent robots, programmed to act like humans – conversational programs that are an application category of AI-based solution, providing a feeling of dealing with a person rather than a computer. The term “*Chatterbot*” was originally coined by *Michael Mauldin* (creator of the first Verbot, Julia) in 1994 to describe these conversational programs. A lot of us would remember “*Microsoft Clippy*”, which is also a form of bot, so to say.

As a form of artificial intelligence, Bots are designed to execute mundane, repetitive tasks that hardly require human intervention and they could practically do everything a software can, such as respond to voice, text and work on files, connect with APIs and other computational tasks and bundled with intelligence, improving productivity.

**Conversation as a Service (CaaS)** and **Conversational User Interfaces (CUI)** has been in the market for a while. Every successful application, irrespective of the interface medium would have one thing in common for sure - a great user experience. With Bots, this doesn't change and hence creating a good conversational user experience is imperative.

And the Bot offerings from Microsoft has made life of developers easier to build, integrate and deploy bots.

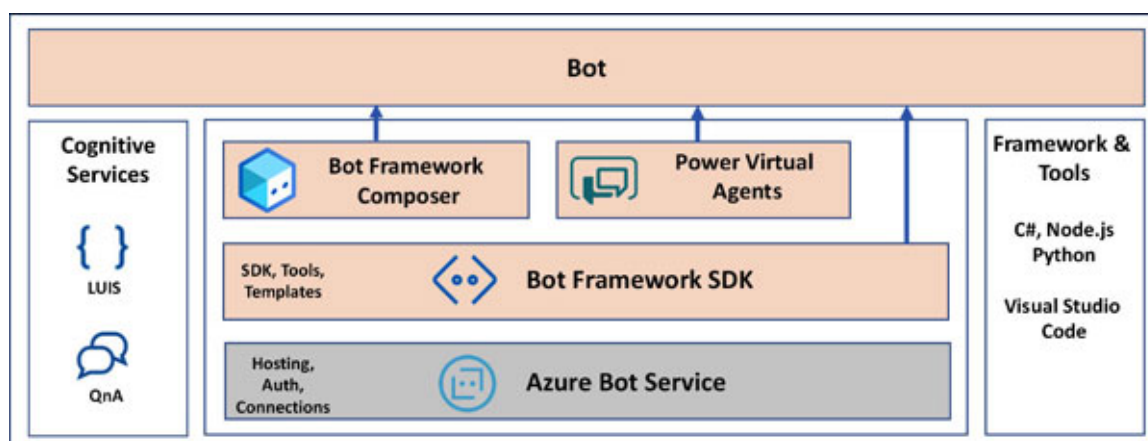
Bots could serve customer scenarios, internal or external. In the internal world, they could empower users to search for knowledge, documents and also seek help by acting as a virtual helpdesk for employees and users. In the external world, Bots can find their way in enterprises not only for simple customer scenarios such as Enquiries, Booking etc., but also advanced ones such as Account management, Payments, Retail etc.

Let us look at the details, we will start with what Bot Framework has to offer, and quickly run through the Emulator along with a demo bot, and we will discuss Bot Framework Composer.

## The Bot Ecosystem

Microsoft offers The Bot Framework as the underlying platform for building bots. Azure Bot Service is an Azure service that helps in hosting bots on Azure and provides features such as connectivity and authentication. It is important to understand this difference between Bot Framework and the Bot Service. In addition to the Bot Framework which is the underlying, Microsoft has other offerings such as the “*Bot Composer*” and “*Power Virtual Agents*” built on top of the Bot Framework that provide readily available interfaces and functions to quickly develop applications.

Here is a quick view of how the Bot landscape looks with the different services made available by Microsoft:



*Figure 6.1: Bot landscape*

## The Bot Framework



Microsoft Bot Framework provides just what you need to build and connect intelligent bots. The Framework implements standard protocols and provides beautiful tools to help model conversations. It also provides UI across multiple canvases and the adaptive cards rock. With Cognitive services integration, it becomes easy to enable Bots with common and well understood patterns and build intelligence into them.

Bots have to be designed keeping the user interface requirements in mind and users have to type or speak as less as possible to get to what they need. Some key considerations while designing a bot are not related to its intelligence but about how better it makes the user experience are as follows:

- **Discoverability:** How easily can customers find the bot and start using it? Does it support multiple platforms?
- **Usability:** Number of steps to address a user's problem.
- **Productivity:** Time taken to solve the situation versus other mediums of communication.
- **Common Patterns:** Bots are great friends to humans, and everyone looks at a use case to bring in bots. Let us take a look at the most common patterns where they get used the most.
- **Task automation:** Bots can perform simple to complex tasks, A simple example could be a user password reset while more complex business processes such as customer order processing are termed as **Robotic Process Automation (RPA)**. Interestingly, the use of software automation to perform tasks has become popular on social media and is called as **Botting**. Some companies provide this as a service to boost the following on social media. Instagram uses neural networks to detect botting by analyzing word frequency in comments posted and accounts get a "*shadowban*" or even deleted.
- **Access to knowledge and other content:** Bots can be designed to provide information about any topic and it can find and return the information that the user has requested by leveraging data from such as relational data in a SQL database, JSON data in a NoSQL store, or documents in a document store. They differ from a normal chatbot, helping in decision making by training a vast amount of unstructured and structured data that produces an expert response.
- **Human hand-off:** Bots could be super intelligent, but there are times when they do not live in isolation and have to hand over the

conversations to humans. The most common reasons for a bot + human hybrid scenarios are situations where it gets complex to handle, or if the situation is critical and needs to be resolved quickly or even simply customer's choice. In such scenarios, there needs to be a hand-off trigger from where a human can take over, with or without wait time involved during the transition.

- **Bot to web browser and back:** There are scenarios where Bots could send a user to a web browser to complete a task and afterwards resume back once the task is done. Most typical use case is Authentication and Authorization with services like Azure Active Directory to access user's personal content such as on Office 365. Performing this redirect to a browser and back is a challenge and often when implementing a channel, it is a good practice to offer a built-in HTML window that the bot can use to present applications that would otherwise appear in a web browser. This keeps the user to within the conversation while still external resources can be accessed.
- **Websites:** Outside of the standard client applications such as Skype, Teams, etc., bots can also exist in a web browser, which is called a "*Web Chat*". Instances like where websites would like to engage customers providing them with Information or Support, can be Bots embedded in the web site and making it easier for users to find information or get support quickly rather than navigate through the site. Bots can then hand-off the conversation to a human after taking an initial stab at the customer request.
- **Apps:** Bots can be integrated with application directly, using the Direct Line API. Be it Native mobile applications, Web applications or custom built applications that run on computers or devices, Bots can embed themselves into these applications and become the first respondents to users and customers.

## [The Bot Framework Composer](#)

There has to be a tool for everyone, and while Bot Framework SDK is for coders, there wasn't an option for non-coders to work with the Bot Framework. With Bot Framework Composer, Microsoft provides an open-source visual designer that helps in visual bot development. This tool was released around mid-2020 and Microsoft's Bot Framework Engineering team has provided a tool that enables rapid developments of Bots, so that one can spend less time

wiring up and provisioning services. Do not confuse this with the “*Power Virtual Agents*” which targets the business users and citizen developers and is a “*No-Code*” platform as well. The difference with Composer is that it still provides you with the robust framework to build bots, a middle ground between the Bot Framework SDK that provides total control in building your bots, and the Bot Composer that simplifies bot development with visual abstraction.

## **Why should you care?**

Built on top of the Bot Framework SDK, Composer IDE helps author bots, and integrate cognitive intelligence services, all in a visual experience. This caters to an audience who don’t want to code but still want to build bots for their domain specific or business use cases using all the bells and whistles that traditional developers has at their disposal. Being open source, doing custom builds or, hosting in cloud as a “*web application*” is possible. You heard that right, composer can run as a web application or as a desktop tool.

The following tasks can be performed using Bot Composer:

- If you have been working with the Bot Framework SDK for quite a while and decide to switch to no-code model and still retain your work, Bot Composer makes it possible. Existing dialogs and skills can be imported to the Composer as well. That’s an advantage that further speeds up bot development when one decides to switch to the Visual Designer. You can also export them and share.
- Composer uses declarative JSON that sits behind the visual canvas where dialogs and the responses can be stitched together.
- It supports NuGet package manager, NuGet packages can be consumed in bots that lights up the ability to reuse content within an organization just like any other application development.
- You can test your bots using Composer’s web chat feature or use the Bot Emulator.

## **Setting up Composer**

Composer desktop application is available on Windows, MacOS and Linux and can be installed from [here](#). Make you are able to connect to the internet. If you are interested in making custom builds of composer, refer this [Microsoft documentation](#) for pre-requirements and guidance.

Following are the steps required to set up the composer:

## Configuration

- Connections

Connections are a way of linking Bots with external services that could either be from Azure or other external sources. Azure connections could include Web Chat, Teams, Speech service etc. and external connections are external extensions available as packages via the NuGet packages manager.

- Skills

Skills allow referencing existing skills can be associated with the bot for re-use.

## Components of Bot Design using Composer

We will explore the following core features of Composer:

- Dialogs
- Recognizers
- Triggers and Actions
- Natural Language Processing
- Skills
- Memory Scopes

## Dialogs

Dialogs are the means to establish and manage conversations. They are called as “*Adaptive Dialogs*”, think of them as a window or interface to engage with Bots with different functionality tailored for the input and response in each of them. Dialogs can have one or more triggers which are starting points of the conversation. A bot created with bot composer, can have two kinds of dialogs – One “*main dialog*” and one or more “*child dialogs*”.

## Recognizers

A recognizer is the one that interprets messages from the user and splits it into the “*intent*” and “*entity values*”. After this process, the message is passed to the

Trigger. There are 3 types of recognizers, the Default recognizer, which is default, enables the use of LUIS and QnA recognizers that come in-built with the composer. It is also possible to have custom recognizers or use the regular expression version that lets custom extraction of intent and entity values.

## Triggers and Actions

A Trigger is an event initiated by a rule written for a given dialog and has one or more actions. An action is an execution instruction for the bot to handle. Example of a Trigger is “*Intent recognized*” and when something the intent, let’s say is “*Agent*”, the action could be to “*Begin a new Dialog*” that starts a separate conversation flow with the agent.

## Natural Language Processing

**Natural Language Processing (NLP)** is a technology that encompasses **Natural Language Understanding (NLU)** and **Natural Language Generation (NLG)**. NLU is the ability to understand decipher text to know what it means. Imagine a user communicating with a bot using words such as <TBD>. NLG is all about generating natural language used to respond to users. Let us look at some details.

There are four NLP components in composer:

- Language understanding
- Language generation
- QnA
- Orchestration

## Language understanding

Interpreting user input is key to establishing a conversational flow when users start their interaction with Bots. Unless the intention of users is understood, it isn’t possible to step into the next action. To interpret the language or text of a user, Language understanding allows to train language models and associate them to the bot. In Composers, this can be done within the dialog framework which allows associating different types of “*recognizers*” such as LUIS, Orchestrator and regular expression recognizers.

## Language generation

Once the intent and utterances are determined, language generation helps in provided responses to the users. This is integrated with the composer’s framework that allows the response editor to create replies that are sensitive to the context of the conversations, correctness of grammar, and so on.

## QnA

QnA maker or the latest Cognitive Service for Language allows to associate knowledge base with bots to provide question-answer style conversation by users. Refer to the QnA section for more details on this topic.

## Orchestration

Orchestrator can be enabled during the creation of the Bot or can be enabled later as well.



*Figure 6.2: Enabling Orchestrator*

## Skills

Skills are used to create shareable conversation logic. A Skill is like a coded function that does something finite – via a task or set of tasks, that can be called from one or more bots. It’s a mini-bot, so to say, since it could be a user-facing bot, and a skill, that can be re-used in other bots.

A skill is described by a skill manifest (JSON file) that defines its actions, input, and output. You can create a skill by exporting a bot with Composer. Bots that make use of a skill are termed as “*root bots*”. When a skill is used by a bot (and not directly used a bot), users can only interact with the “*root bot*” and not with the skill directly.

## Memory Scopes

To maintain the conversational flow, bots need to save the state of the user conversation and context, so that users can continue from where they left-off. By default, active dialogs are saved to the memory, and other data such as user profiles, preferences, custom properties can also be added. All this is done via the properties pane where you could try properties that can be used for Bot response, user input and custom.

## Cognitive Intelligence

Building an intelligent bot requires stitching together several components. Microsoft's Conversational AI tools including the Bot Service, LUIS, QnA Maker, and other cognitive intelligence services such as language, speech, and semantic understanding etc., enables developers to build, and deploy intelligent bots that naturally interact with users on multiple channels.

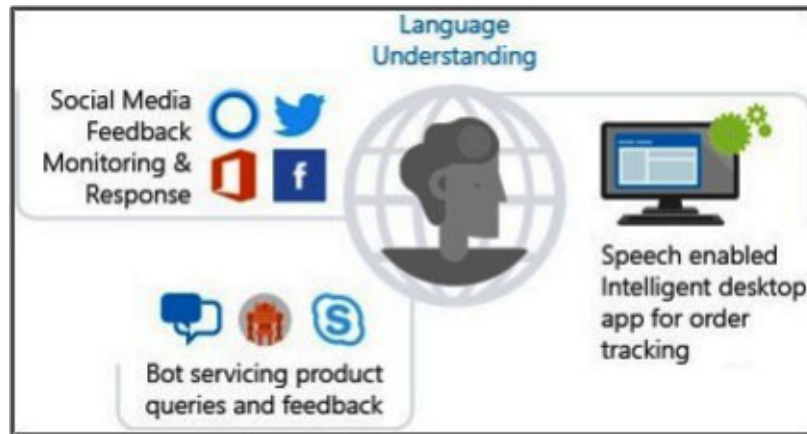
For instance, the Microsoft Research project "*Project Conversation Learner*" enables building and teaching conversational interfaces that learn directly from interactions. It applies machine learning behind-the scenes to decrease manual coding of dialogue control logic and its SDK works in conjunction with the Bot Builder SDK.

### QnA Maker

QnA Maker (Managed) is the classic question and answer service that provides question answer based conversational dialogue capability. This is available separately under the Cognitive Services and is an updated version of the erstwhile "*QnA Maker*" service. The QnA Maker Managed update brings simplified resource management by creating just two resources as part of the service namely the QnA Maker Service and the Cognitive Search. It also has support for Azure Monitor for monitoring (as against Application Insights that was available before) and provides deep learning NLP ranking model as well. We will use this classic service in our sample using the Bot Framework SDK later in the chapter.

### LUIS - Language Understanding Intelligent Service

Microsoft designed LUIS with a vision of democratizing machine learning. A few years ago, building your own machine learning models required data scientists who spent hours and days and sometimes maybe years in order to build and customize "*machine learned*" models for specific domains.



*Figure 6.3: LUIS Enabled Apps*

A LUIS app can be created using the [luis.ai](https://luis.ai) portal and such an app contains a domain specific natural language model. If you are starting new, you should start off by trying out one of the prebuilt domain model. The other way is to build your own model, and it is also possible to have blended combination of prebuilt domain models along with your custom defined model.

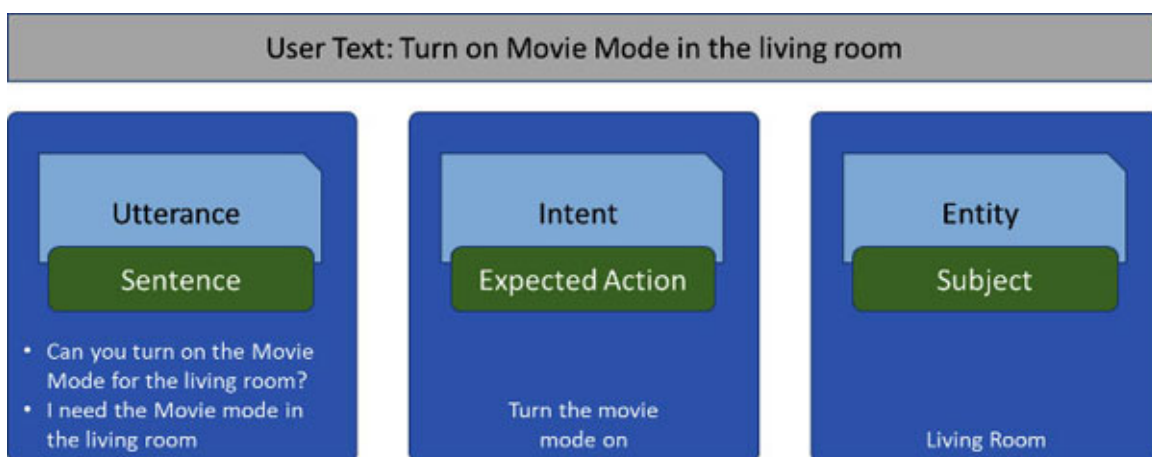
The following are important concepts to understand in a LUIS application:

**Intent:** Action users wants the bot to perform - the basic intention behind user's query.

**Utterances:** Questions or sentences used by users with the bot.

**Entities:** Nouns or Objects in the context that the users' query is based on.

Here is an example:



*Figure 6.4: Main Components of LUIS App*




The way LUIS is different from QnA Maker, is primarily on how the API response is consumed. With QnA Maker, the result of the query, which is the answer to a mapped question is directly sent to the user, with no additional processing or handling by the bot (bot's code). With LUIS, response from the LUIS API is a combination of prediction score, intent, and some extra metadata. Bot must handle this response and react accordingly before it can respond to users.



## [Azure Cognitive Services for Language](#)


In May 2021, Microsoft re-introduced QnA Maker managed service as “*Custom Question Answering*” feature as part of Text Analytics. This was part of the larger “*AI at Scale*” approach, where “*Question Answering*” feature was introduced with both pre-built and custom capabilities. The existing QnA Managed resources continue to work as well.

What Microsoft has done very recently is to unify QnA Maker and LUIS and place it along with text analytics in a new service called as the “*Language Service*” or “*Cognitive Services for Language*”. This is in public preview as of this writing. If you would have QnA or LUIS resources already, it is possible to migrate them to the new language service. Importing LUIS is quite simple by importing your LUIS JSON file, while QnA migration is a guided process. If you are interested, you can take a peek at the migration guide here - [Migrate QnA Maker knowledge bases to custom question answering - Azure Cognitive Services | Microsoft Docs](#)




 Microsoft Azure


[Home](#) > [Cognitive Services](#)


 **Cognitive Services** | Language service 

 Search (Ctrl+ /)


<<


 Create  Manage view 


 Overview

 All Cognitive Services


Decision


 Anomaly detector


 Content moderator


 Personalizer

Language


 Language service

 Translator


 Language understanding (classic)


 QnA maker (classic)


Speech

 Speech service

Vision

 Computer vision

 Custom vision

 Face API

Filter for any field...

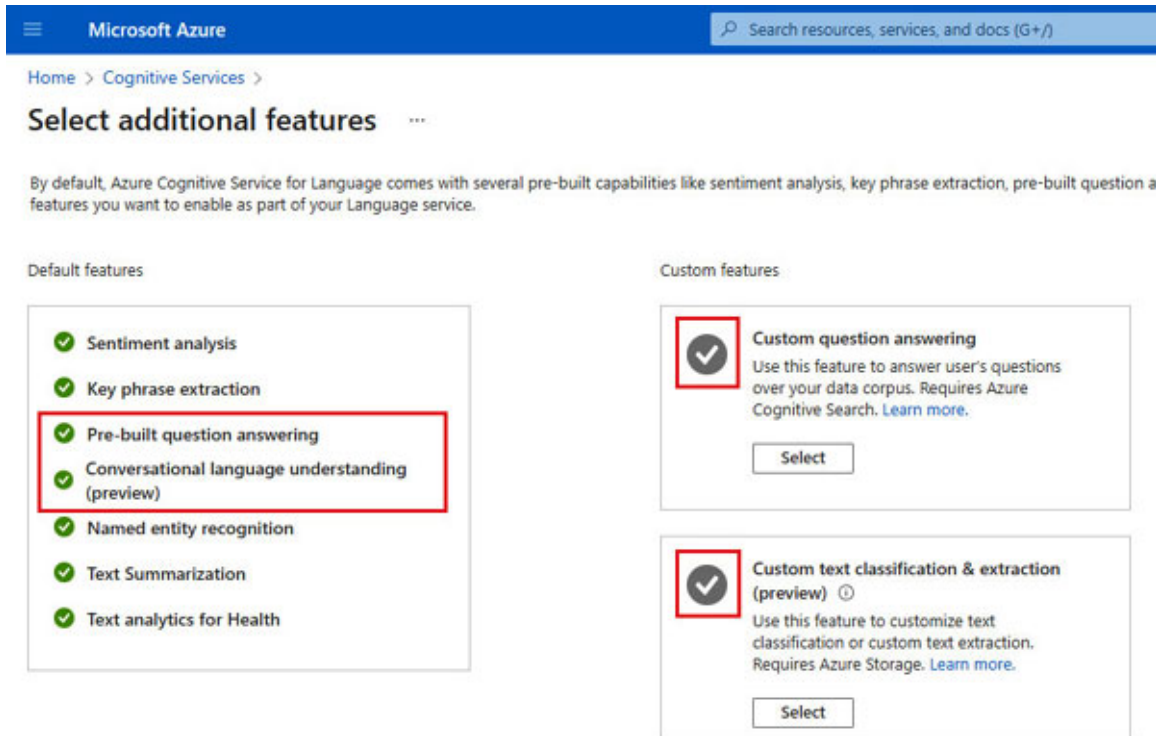
Subs...

Showing 0 to 0 of 0 records.

Name

↑↓

**Figure 6.5:** Cognitive Services – Language Service



*Figure 6.6: Adding Features in a Language Service*

## Search

Search is an integral part of building bots. QnA Maker for instance, now uses Azure Search internally (you would see Azure Search instance created as part of the QnA Service) where Search is used to index all the knowledge bases content and as well as for authoring and testing purposes.

Over and above this, Bots can incorporate Azure Search separately to explore multiple sources involved and could also have Bing Search integrated for web search. Therefore, if the user query could not be serviced by the models and KBs, and the data from the data sources, bots can run a Bing Search and provide with the suitable answer.

## Storage

Storing data with Bots can be of two ways - the in-memory store or database store. In both cases, bots store and retrieve state data that is associated with a user/ conversation and this data can then be used for many purposes, such as to re-start the conversation from where it was left-off or simply greeting a returning user by their name. It can also be used to interact with users with

content tailored to their interests, such as alerting them on an event or news that interests them.

## **Information sources**

Bots can also interface with external informational sources that are exposed as REST APIs. Such services are those that could provide demographic and geospatial data, data from 3rd parties and agencies, and so on.

- **Monitoring and Insights**

Bot Analytics provides analytical insights into the bot. It is an extension of application insights and provides data about traffic, latency, and integrations and also the conversational data such as user, message, and channel data.

- **Reporting/Visualization**

Reporting and visualization is a common aspect of any solution and can be fulfilled by cloud native offerings such as Power BI. Power BI is a collection of software services, apps, and connectors that work together to turn unrelated sources of data into visually immersive, and interactive insights. Data can be in any source such as spreadsheets, cloud based or on-premises based data stores and warehouses.

## **Build Bots using the Bot Framework**

Building Bots using the Bot Framework involves the following components and tools that assist developers in building Bots:

- **Bot Framework SDK:** A dev kit that provides all the necessary plumbing code to develop Bots.
- **Bot builder tools:** To assist with end-to-end development, including the Bot emulator.
- **Azure Bot Service:** The core service that enables communication between bot and its channels.
- **Conversation experiences:** Channels that allow communication with bots.
- The Azure Bot Service provides two ways to create a bot and they differ in terms of how customers are billed which is driven by the requirements/usage of the bot.

- **Web App Bot:** This type of bot uses an App Service to host the bot and has the option of choosing a “*App Service Plan*” or a “*Consumption Plan*”
- **Functions Bot:** Function bots use Azure Functions as the backed. You will not see an App service plan/Location field when creating a function bot. Instead, a Hosting plan field is available to choose.

The thought process of designing and building bots is important even before you start working on design. As a rule, the following should be goals:

Start simple and add complexity as you build on. Bot should be contextual and adapt to the user.

Incorporate intelligent controls to manage complexity.

## Lifecycle

The goals for building a bot is:

- Start simple and add complexity as you build on. Bot should be contextual and adapt to the user.
- Incorporate intelligent controls to manage complexity.

The end to end development workflow for a bot includes 6 stages and Microsoft provides the tools for all of them to help develop, design, and build bots:

Picture courtesy: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0>



*Figure 6.7: End to End Development Life Cycle*

When you start building Bots, the first one you build you are likely going to face a few glitches and feel that you missed to plan it well. But understanding the development lifecycle helps developing a bot from start to finish without many glitches. Here is a quick summary of each phase in the lifecycle.

- **Plan**

Involves planning the bot's conversations and dialogs by leveraging the design best practices. (look at principles of bot design). Use Chatdown to mock conversations before you start coding.

- **Build**

Build phase is when you would tie-in intelligence features such as **Language Understanding (LUIS)**, Knowledge base support (QnA Maker).

- **Test**

Use the Bot Framework Emulator V4 for testing the bot and debugging it with breakpoints. The Emulator is a free tool provided by Microsoft and allows connecting the bot to other services such as LUIS, QnA, and so on.

- **Publish**

Bots can be published directly from Visual Studio or VS Code if you use them for custom development. Azure CLI can also be used to create, download and publish your locally built bots to the Azure Bot Service.

- **Connect**

Channels are the connection between the bot and communication apps. Bot Service Channels can be connected to bots via Azure Portal or using Azure CLI command line.

- **Refine**

Post deployment, evaluation involves improving the capabilities and performance of the Bots. Use Bot Analytics and tools like Application Insights to access conversation-level reporting on user, message and channel data.

## Reference Architecture

A reference architecture shows the possible components that could be used to design a solution, while the actual real-world solution for a business problem may not use all of the services, or may even incorporate additional services that aren't documented in the reference architecture. This reference architecture for Bots depicts the use of Azure Bot Framework and other important services around it that builds a complete solution.

Every bot is unique, yet there is always a most fundamental perspective that can result in any kind of use case – a chat bot, conversational bot, support bot,

etc. This reference architecture should be a good place to start exploring the possibilities and expand them further as requirements evolve.

The architecture portrays the following key pillars:

- **Users**

These are the users - internal staff or customers who are users of the bot via various channels. Users who prefer to use bots are not expected to understand anything about Bots and can converse in natural language of choice by starting off with a Hi or even a pressing question for which they have reached out. Bots guide users via “*guided conversations*”, helping them to traverse the whole conversation by providing options, and helping them choose one. The goal provide users the information/support they need in the best possible time.

- **Azure Bot Service**

Azure Bot Service enables you to host intelligent, enterprise-grade bots with complete ownership and control of customers data. Ideally speaking, you could host bots anywhere you want, but the Bot Service in itself enables users to create bots from existing templates, host bots in Azure and providing connectivity options to connect to various channels, test the bot in a default “*Web Chat*” etc., amongst other features.

- **Conversational Channels**

Channels are various interfaces via which the bot service can be connected to. Some of the common communication channels are Skype, SMS, and email.

- **Security**

Prior to version 4, Bot framework required inclusion of OAuth controllers to perform user token management where the bot would ask the user sign in on a website, and use the magic code the user can verify their identity against. With v4 SDK, there is no need for magic code verification and it is now easier to develop a bot that authenticates users to various identity providers, such as Azure AD or even other provides such as GitHub etc. All of the security is taken care within the Azure Bot Service where you would have to create an Azure AD Application and register the app with your bot.

Some of the best practices should be followed with respect to security are:

Enabling enhanced authentication options in the Direct Line channel - that allows Bots service to detect and reject user id changes thereby avoiding impersonation. Specify the trusted domains in the Direct Line configuration page.

## **The Bot Framework Emulator**

Making development much more enjoyable comes with some powerful tools at developer's disposal and Bot Emulator is one of them. The Bot Framework Emulator is an open-source, cross-platform application that allows developers to build, test and diagnose their conversational applications on local machines or in the cloud. While the emulator doesn't cover all the areas in the workflow outlined above, what it does cover pretty well, is planning, testing and refining Bots and it does an excellent job at that.

The emulator works with both local and remote hosted bots and supports the three major development platforms – Windows, Linux and OS X

The latest version of the emulator is available for download at.

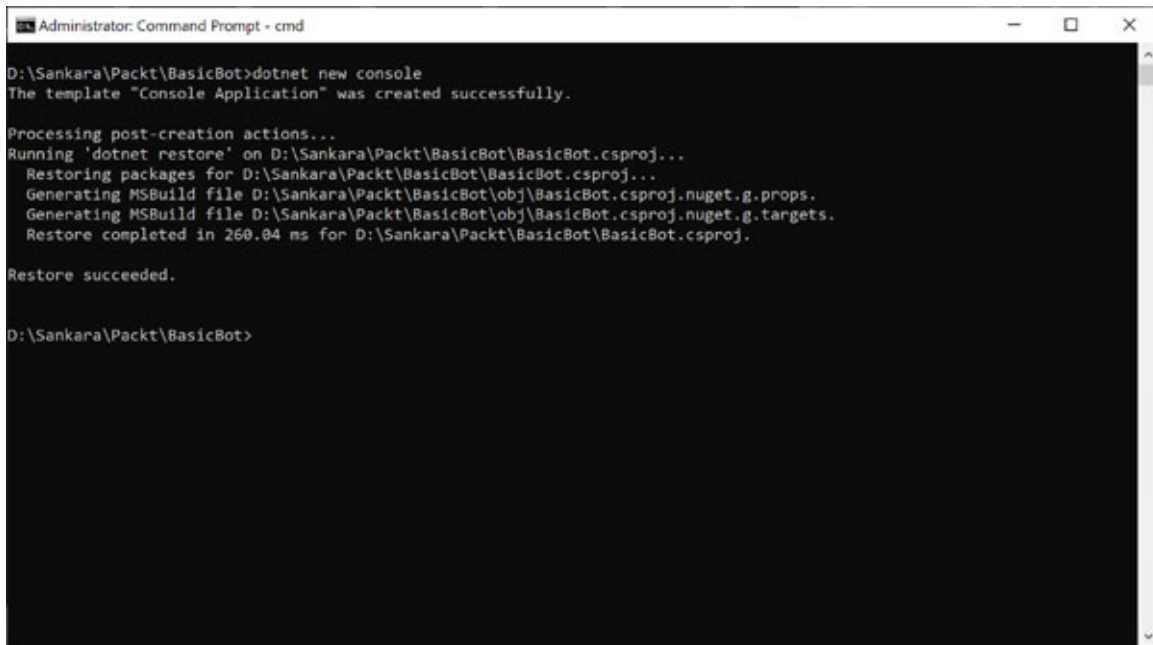
## **Building a Basic Bot**

Let us start with a Basic Bot to demonstrate how it is done and then later we would use the Emulator to debug and test it.

You can use the full version of Visual Studio/ Visual Studio Community or VS Code.

Start off by going to Command Line or PowerShell and create a new project under a new folder “**Basic Bot**” by using the dotnet command line





```
Administrator: Command Prompt - cmd
D:\Sankara\Packt\BasicBot>dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on D:\Sankara\Packt\BasicBot\BasicBot.csproj...
  Restoring packages for D:\Sankara\Packt\BasicBot\BasicBot.csproj...
    Generating MSBuild file D:\Sankara\Packt\BasicBot\obj\BasicBot.csproj.nuget.g.props.
    Generating MSBuild file D:\Sankara\Packt\BasicBot\obj\BasicBot.csproj.nuget.g.targets.
  Restore completed in 260.04 ms for D:\Sankara\Packt\BasicBot\BasicBot.csproj.

Restore succeeded.

D:\Sankara\Packt\BasicBot>
```

*Figure 6.8: Creating a BasicBot Folder*

This creates a project in the **BasicBot** folder.

## Creating a QnA KB

Let us try build QnA Maker into the basic bot, that could handle a question and answer session with a user. We are choosing the classic model here and you should see a warning that asks you to switch to Question Answering - lets ignore it. We would also use the emulator to quickly run through the features and how to test the bot locally before it gets deployed to Azure.

Let's start off by going creating a knowledge base.

- a. Go to *QnAMaker.ai* and sign in with your Azure account
- b. Click on “**Create a knowledge base**” to create a knowledge base which should open the following screen:

Cognitive Services
QnA Maker
My knowledge base
Create a knowledge base
7

## Create a knowledge base

Create an Azure service for your QnA knowledge base and add sources that contain the question and answer pairs you would like to include. [Learn more about creating a knowledge base.](#)

STEP 1

### Create a QnA service in Microsoft Azure.

Create an Azure QnA service for your KB. If you already have an Azure QnA service for this KB, skip this step. Select "Preview" to try Custom Question Answering (preview release feature). [Learn more about Azure subscriptions, pricing tiers, and keys.](#)

Create a QnA service

We strongly recommend you to use the Question Answering feature (new release). Available on the Azure Cognitive Service for language. You can add language models to create a language service.

STEP 2

### Connect your QnA service to your KB.

After you create an Azure QnA service, refresh this page and then select your Azure service using the options below.

Refresh

Microsoft Azure Directory ID

Search...

Azure subscription name

Select subscription

Azure QnA service

Select service

Select language

STEP 3

### Name your KB.

The knowledge base name is for your reference and you can change it at anytime.

Name

Name your knowledge base

STEP 4

### Populate your KB.

Extract question-and-answer pairs from an online HTML product manual, or other files. Supported formats are .txt, .pdf, .doc, .docx, .xlsx, containing questions and answers in sequence. [Learn more about knowledge base sources.](#) Skip this step to add questions and answers manually after creation. The number of sources and the size you can add depends on the QnA service SKU you choose. [Learn more about QnA Maker SKUs.](#)

☐ Enable multi-turn extraction from URLs, .pdf or .doc files. [Learn more.](#)

URL

https://

+ Add URL

File name

+ Add file

Chat chat

Give your bot the ability to answer thousands of chat-bot questions in a voice that fits your brand. When you add chat-bot to your knowledge base by selecting a personality below, the questions and responses will be automatically added to your knowledge base, and you'll be able to edit them anytime you want. [Learn more about chat-bot.](#)

☒ None

☐ Professional

☐ Friendly

☐ Witty

☐ Caring

☐ Enthusiastic

STEP 5

### Create your KB

The tool will look through your documents and create a knowledge base for your service. If you are not using an existing document, the tool will create an empty knowledge base (one which you can edit).

Create your KB

Terms of Use
Guide of Content
Feedback
© 2022 Microsoft

**Figure 6.9:** Creating a Knowledge base

The page provides with a step by step approach towards creating the knowledge base, and follows-up to create a bot:

# Create a knowledge base

Create an Azure service for your QnA knowledge base and add sources that contain the question and answer pairs you would like to include.  
[Learn more about - creating a knowledge base.](#)

## STEP 1

### Create a QnA service in Microsoft Azure.

Create an Azure QnA service for your KB. If you already have an Azure QnA service for this KB, skip this step. Select 'Preview' to try Custom question answering (preview release) feature. [Learn more about Azure subscriptions, pricing tiers, and keys.](#)

Create a QnA Service

① We strongly recommend you to use the Question Answering feature (now Generally Available) within Azure Cognitive Service for Language. You can visit [Language studio](#) to create a Language resource.

## STEP 2

### Connect your QnA service to your KB.

After you create an Azure QnA service, refresh this page and then select your Azure service using the options below

Refresh

\* Microsoft Azure Directory ID

merkato

\* Azure subscription name

Select subscription

\* Azure QnA service

Select service

*Figure 6.10: Creating a QnA service*

Click on “**Create a QnA service**” button which will take you to the “**QnA Maker**” creation blade under Cognitive Services, in the Azure Portal.

Home > Cognitive Services >

## Create

QnA Maker

Basics Tags Review + create

QnA Maker is a cloud-based API service that lets you create a conversational question-and-answer layer over your existing data. Use it to build a knowledge base by extracting questions and answers from your semi-structured content, including FAQs, manuals, and documents. Answer users' questions with the best answers from the QnAs in your knowledge base automatically. Your knowledge base gets smarter, too, as it continually learns from user behavior. [Learn more](#)

**i** QnA Maker managed (preview) is now a Generally Available feature within Azure Cognitive Service for Language, and it has been renamed to custom question answering. [Create a Language resource](#) to use question answering and other features such as entity recognition, sentiment analysis, etc.

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Visual Studio Enterprise Subscription
Resource group *	(New) AIBook
	<a href="#">Create new</a>
Resource group location *	(US) West US
Name *	BasicBot-QnA
Pricing tier ( <a href="#">Learn More</a> ) *	Standard S0 (\$10 per month for unlimited documents, 3 transactions...

### Azure Search details - for data

When you create a QnAMaker resource, you host the data in your own Azure subscription. Azure Search is used to index your data.

Azure Search location *	(US) West US
Azure Search pricing tier *	Free F (3 Indexes)

### App Service details - for runtime

When you create a QnAMaker resource, you host the runtime in your own Azure subscription. App Service is the compute engine that runs the QnA Maker queries for you.

App name *	BasicBot-QnA-ab59
Website location *	(US) West US

**i** The App service plan currently defaults to standard(S1) tier ([Pricing](#)). It can be modified by visiting the app service plan resource page once the resource has been created.

### App insights details - for telemetry and chat logs

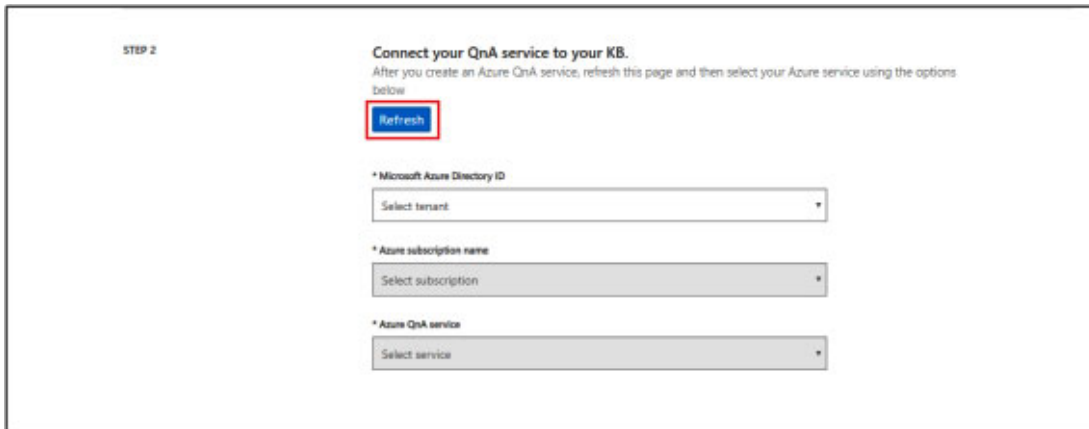
QnAMaker will optionally provision an instance of Application Insights and will appear in your Azure subscription. Telemetry and chatlogs will be stored here.

App insights	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
App insights location *	(US) West US

**Figure 6.11:** Creating a QnA service

Choose the desired options, for the context, we choose the basic pricing tiers for both QnA and Search. and once created, user is redirected to the overview page. You could explore the options in here and the most important information for us would be the “Keys” blade which has the keys for use with Bot Service.

Back in the QnA Maker portal, hit the “*Refresh*” button in Step 2 to see your newly created QnA Service listed and selected the service to connect the knowledge base to the Azure based service. QnA Maker is a SaaS tool/offering and hence it (and its knowledge base) is separated out from Azure. The Azure QnA Maker service is a billing entity.



STEP 2

**Connect your QnA service to your KB.**  
After you create an Azure QnA service, refresh this page and then select your Azure service using the options below

**Refresh**

\* Microsoft Azure Directory ID  
Select tenant

\* Azure subscription name  
Select subscription

\* Azure QnA service  
Select service

**Figure 6.12:** Connecting to Database

Give a name to the knowledge base and provide the FAQ or attach a document to automatically have the data extracted and populated from the reference link or document. You will be provided with the option to make edits before the information can be published for use.

STEP 3

#### Name your KB.

The knowledge base name is for your reference and you can change it at anytime.

\* Name

Name your knowledge base

STEP 4

#### Populate your KB.

Extract question-and-answer pairs from an online FAQ, product manuals, or other files. Supported formats are .tsv, .pdf, .doc, .docx, .xlsx, containing questions and answers in sequence. [Learn more about knowledge base sources.](#) Skip this step to add questions and answers manually after creation. The number of sources and file size you can add depends on the QnA service SKU you choose. [Learn more about QnA Maker SKUs.](#)

☐ Enable multi-turn extraction from URLs, .pdf or .docx files. [Learn more.](#)

URL

<https://www.mcdonalds.com/us/en-us/faq/burgers.html>

**Figure 6.13: Populating the KB**

The last part before you go ahead and create the KB is to select the optional Chit-chat personality, which helps the bot to be more conversational and engaging and defines the personality of the virtual assistant.

+ Add URL

File name

+ Add file

#### Chit-chat

Give your bot the ability to answer thousands of small-talk questions in a voice that fits your brand. When you add chit-chat to your knowledge base by selecting a personality below, the questions and responses will be automatically added to your knowledge base, and you'll be able to edit them anytime you want. [Learn more about chit-chat.](#)

- ☒ None
- ☐ Professional
- ☐ Friendly
- ☐ Witty
- ☐ Caring
- ☐ Enthusiastic

STEP 5

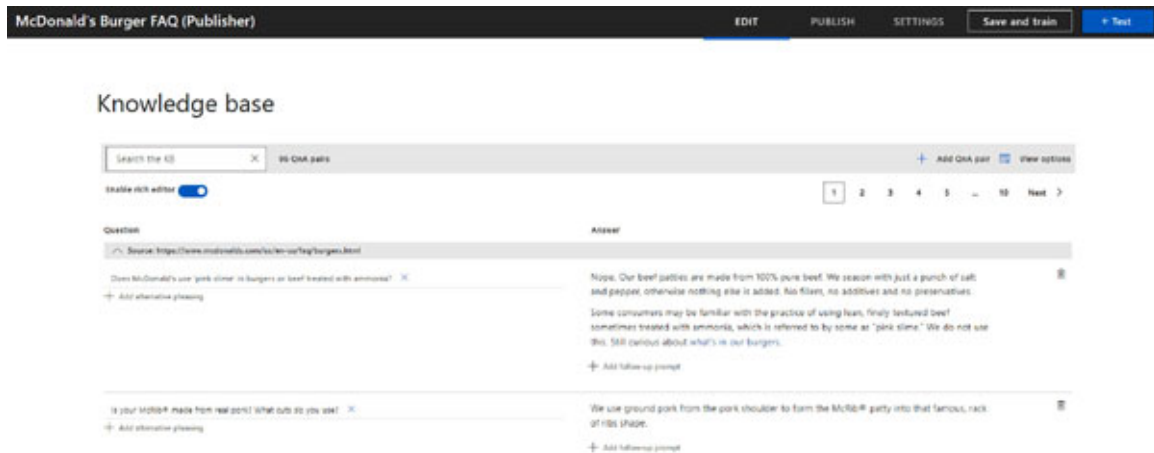
#### Create your KB

The tool will look through your documents and create a knowledge base for your service. If you are not using an existing document, the tool will create an empty knowledge base table which you can edit.

Create your KB

**Figure 6.14: Chit Chat Functionality**

Knowledge base gets created and is displayed to the user, and this is the time they could review, make changes. Once done, the information can be saved and trained using the “**Save and train**” button and later Published using the “**Publish**” tab. You can also test the knowledge base within the portal and ascertain its functioning before publishing it.



*Figure 6.15: Publish the Service*

Once the knowledge base gets published, you are now ready to create a bot and it is now

integrated within QnAMaker with a “**Create Bot**” button.

**Success! Your service has been deployed. What's next?**

You can always find the deployment details in your service's settings.

**Create Bot**

[View all your bots on the Azure Portal.](#)

Use the below HTTP request to call your Knowledgebase. [Learn more.](#)

Postman    Curl

```
POST /knowledgebases/4401a4c3-07f1-41ec-94ce-4277e6046549/generateAnswer
Host: https://basicbot-qna-9df7.azurewebsites.net/qnamaker
Authorization: EndpointKey be46873f-43a4-4913-a199-b56664b74950
Content-Type: application/json
{"question": "<Your question>"}
```

Need to fine-tune and refine? Go back and keep editing your service.

**Edit Service**

*Figure 6.16: Chit Chat Functionality*

Clicking the button smartly opens up the Azure Web App Bot blade with all necessary information pre-populated, which otherwise is a difficult task of noting down or remembering





[Home](#) >

# Web App Bot ...

Bot Service

Bot handle \* ⓘ

basicbot-qna-9df7-bot

Subscription \*

Visual Studio Enterprise Subscription

Resource group \*

AzureAIBook

[Create new](#)

Location \* ⓘ

West US

Pricing tier ([View full pricing details](#)) \*

F0 (10K Premium Messages)

App name \* ⓘ

basicbot-qna-9df7-bot-af5b

.azurewebsites.net

SDK language \*

☒ C# ☐ Node.js

QnA Auth Key \* ⓘ

be46873f-43a4-4913-a199-b56664b74950

App service plan/Location \*

BasicBot-QnA/West US



Application Insights ⓘ

☒ On

Application Insights Location \* ⓘ

West US

Microsoft App ID and password ⓘ

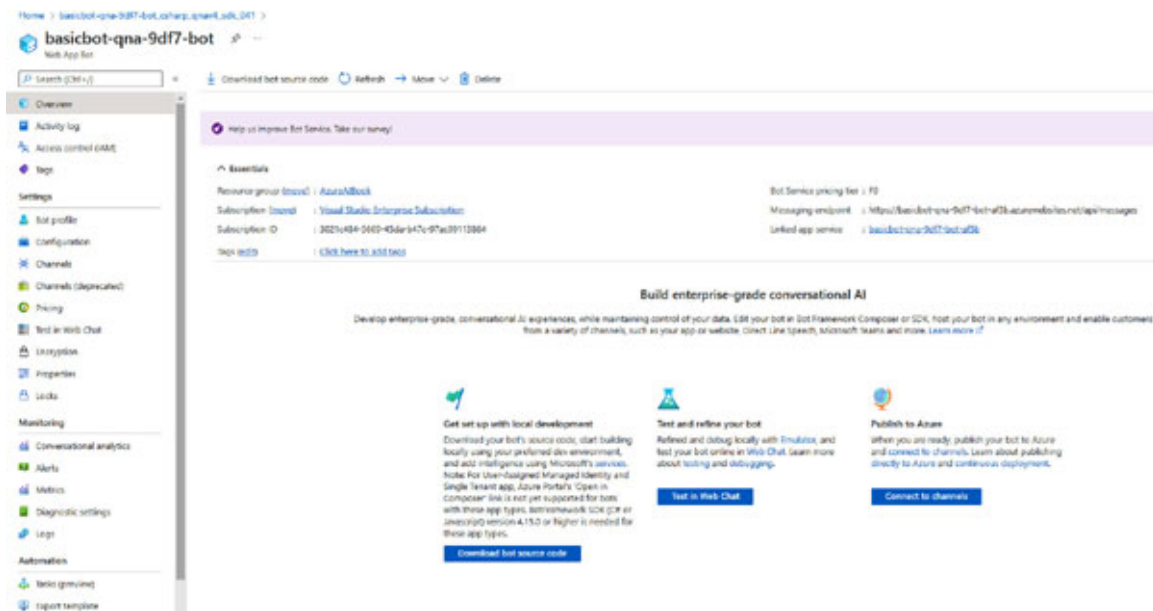
Auto create App ID and password



Create

*Figure 6.17: Create Bot*

Click on create and wait for the bot to be created. At this stage, you are ready to use the bot backed by QnA Maker's knowledge base filled with McDonald's Burger FAQ data beneath it. If you wish to download the code for further changes or to add on additional services, which is usually the case when you develop a bot, go to the "Build" blade and click on the "Download bot source code" in the overview pane and proceed to download the code when the download is ready.



*Figure 6.18: Basic Bot*

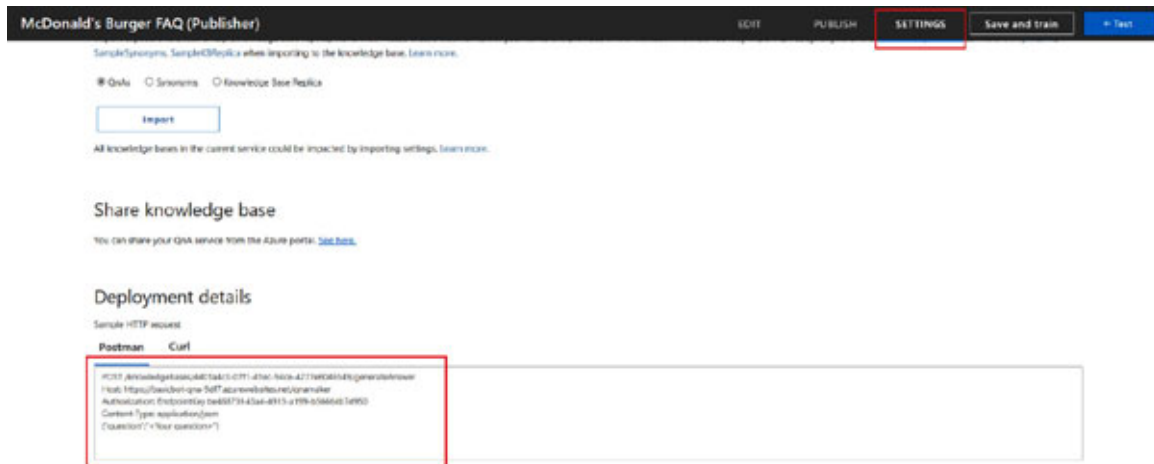
## Using the Emulator

Once you have the code and want to test in a local environment, then is when you the Bot Framework Emulator comes to use. The downloaded code can be opened up in Visual Studio or VS Code for further development and build, or just if it's just build and run, then command line or PowerShell can be used too.

When the code is executed and the bot is running, it listens on the default port - 3879 (localhost). Open up the Emulator and connect to the running bot to interact and test its working.

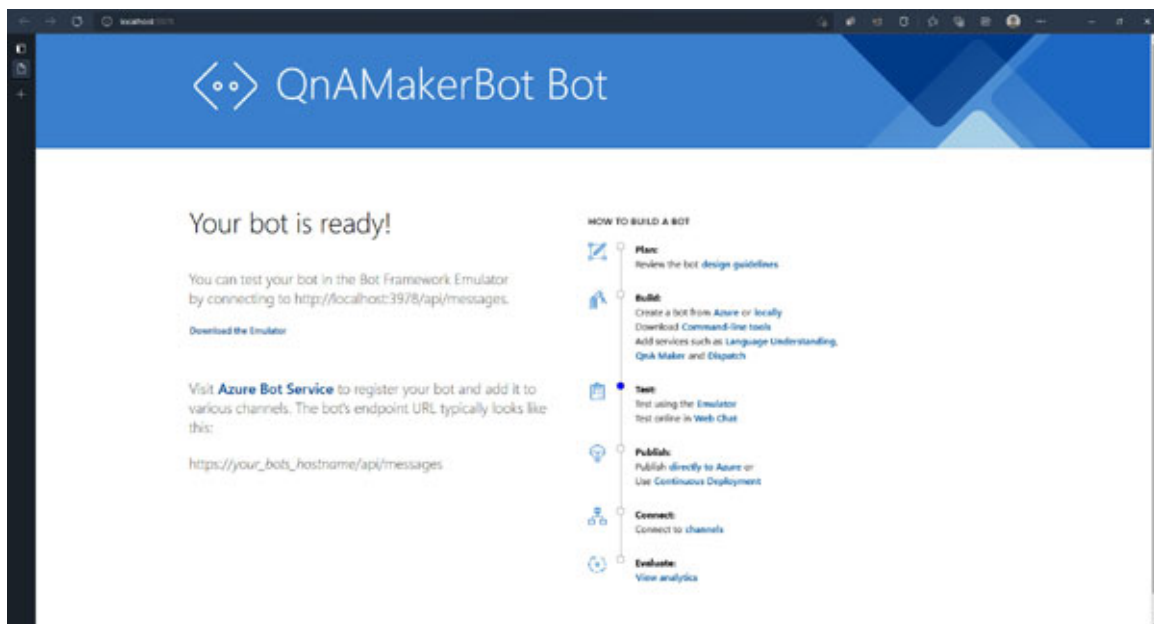
As of this writing, the latest release of the Bot Framework emulator is 4.14.1 released on 21-Nov-2021.

You will need to update the `appsettings.json` file in the downloaded code with the details such as the KB Id, Endpoint, Key etc. You can get them from under “*settings*” of your Knowledge Base.



**Figure 6.19: Bot Emulator**

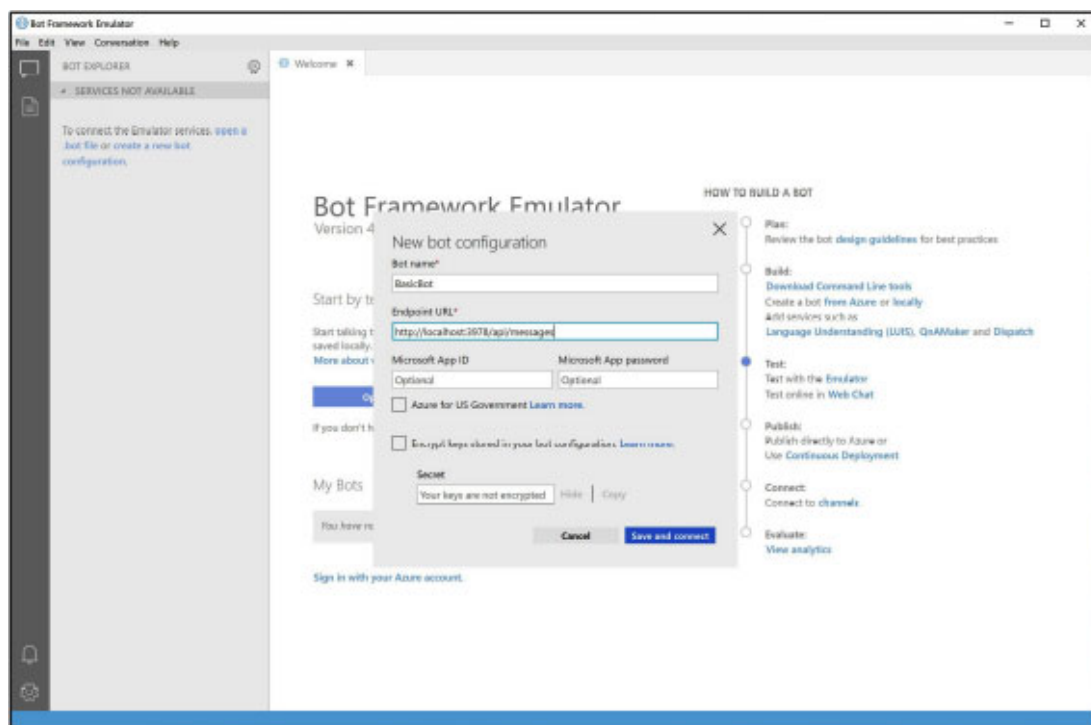
After you update, you can run the bot either via Visual Studio or via Command line and you should the bot should come up in your browser.



**Figure 6.20:** Basic Bot which is ready

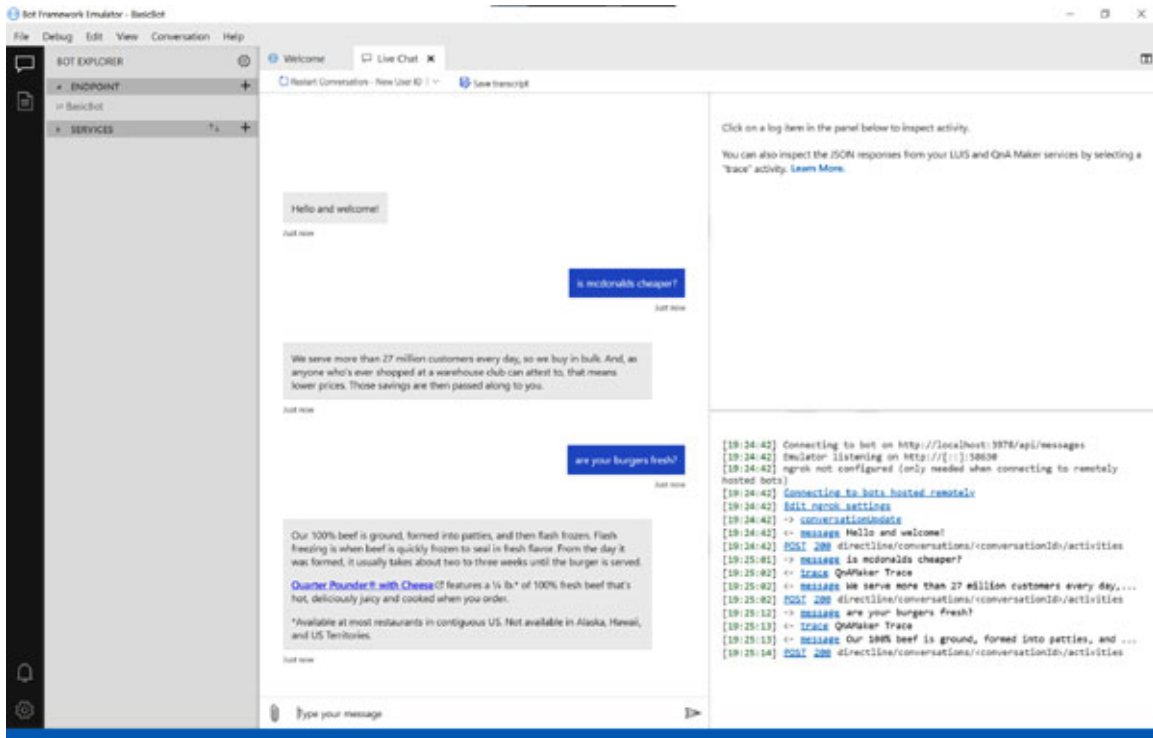
Open the emulator the connect to the bot. If you got an existing bot configuration file, it can be opened up straightaway using the “**Open Bot**” button or via the file menu, and you are ready to run or debug the bot. If it is

the first time, then you would create a new bot configuration using the **File** menu and click on “**New bot configuration**”.



*Figure 6.21: Bot Emulator*

Once the endpoint is created, emulator makes a connection and is now ready to go. You could start conversing with the bot and here is a sample chat that shows responses retrieved from the QnA Maker created earlier based on McDonald’s FAQ.

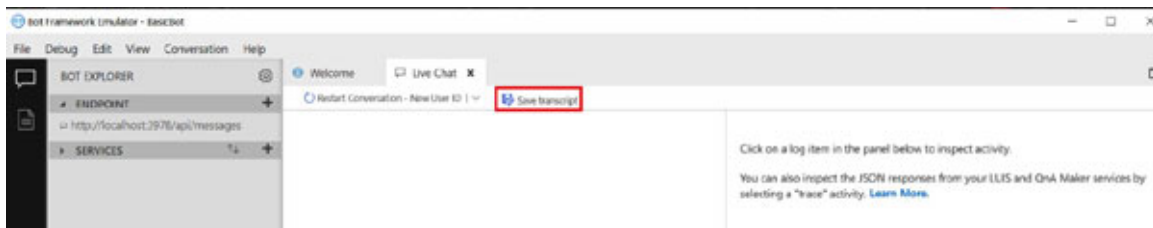


*Figure 6.22: Sample chat that shows responses retrieved from the QnA Maker created earlier based on McDonald's FAQ.*

## Features

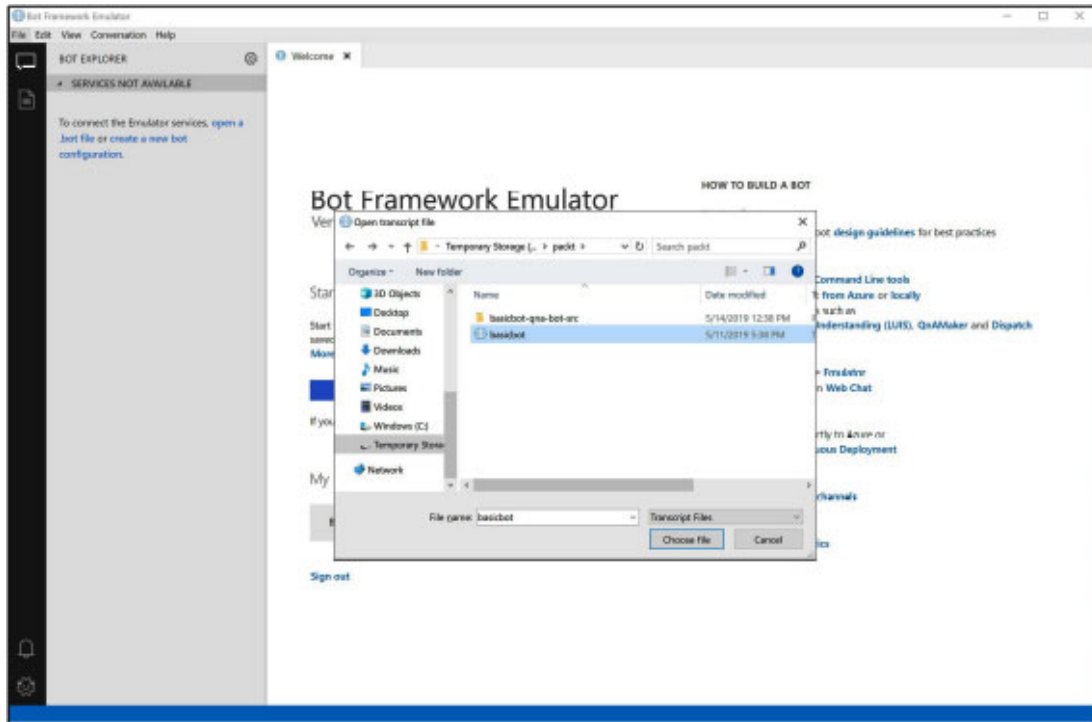
It is important to understand the features and concepts of the Emulator to effectively make use of it, let us take a spin.

**Transcripts:** A bot transcript file is a JSON file that preserves the user-bot interactions. A transcript file not only preserves the message's content, but also interaction details such as user Id, channel Id, type of channel, channel capabilities, interaction time, and so on.



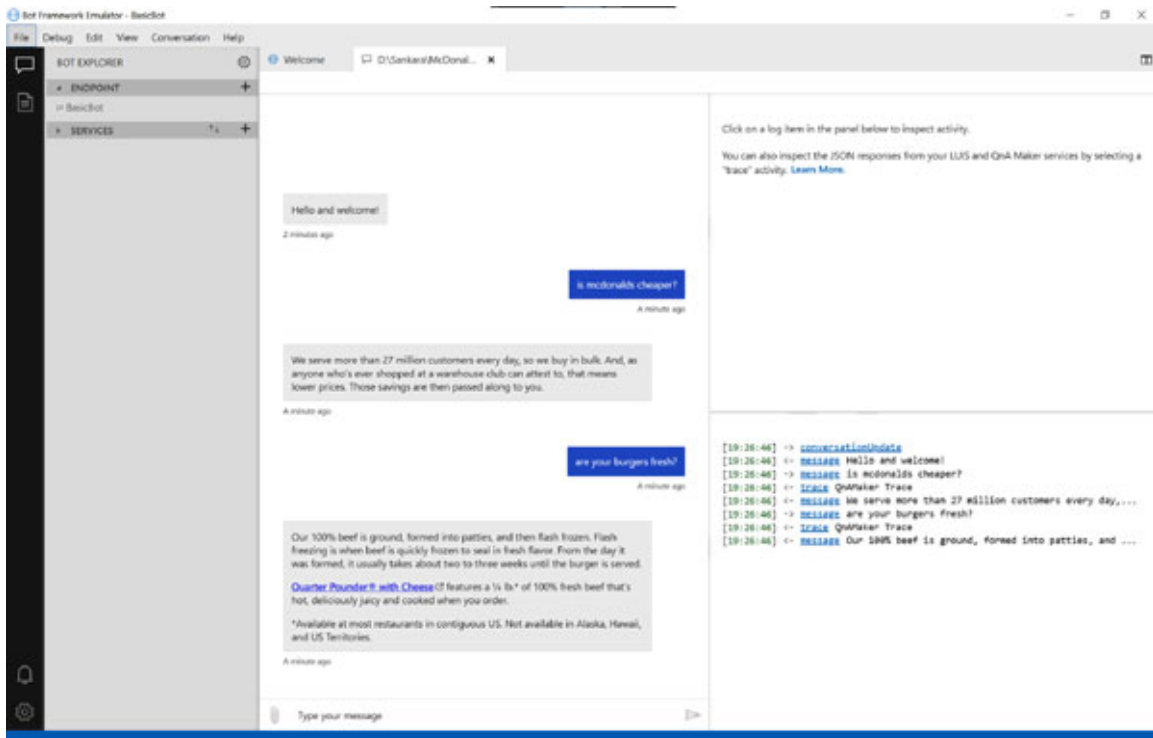
*Figure 6.23: Transcripts*

Saved transcripts can be opened again from the **File** menu | **Open Transcript** option.



*Figure 6.24: Saved Transcripts*

And it displays the entire conversation along with the log.



*Figure 6.25: Conversation with logs*

**Chat Files:** Chat Files are used to create mockups of conversations between a user and the bot. This is especially useful when the conversation has to be created and tested for different paths (conversations), or if it has to be discussed or reviewed with non-technical stakeholders. A chat file has `.chat` extension and looks like the following:

```
user=sankara bot=BasicBot user: Hi!  
bot: Hello there, I am the basic bot. How can I help you today?
```

It basically is a markdown file that contain 2 parts:

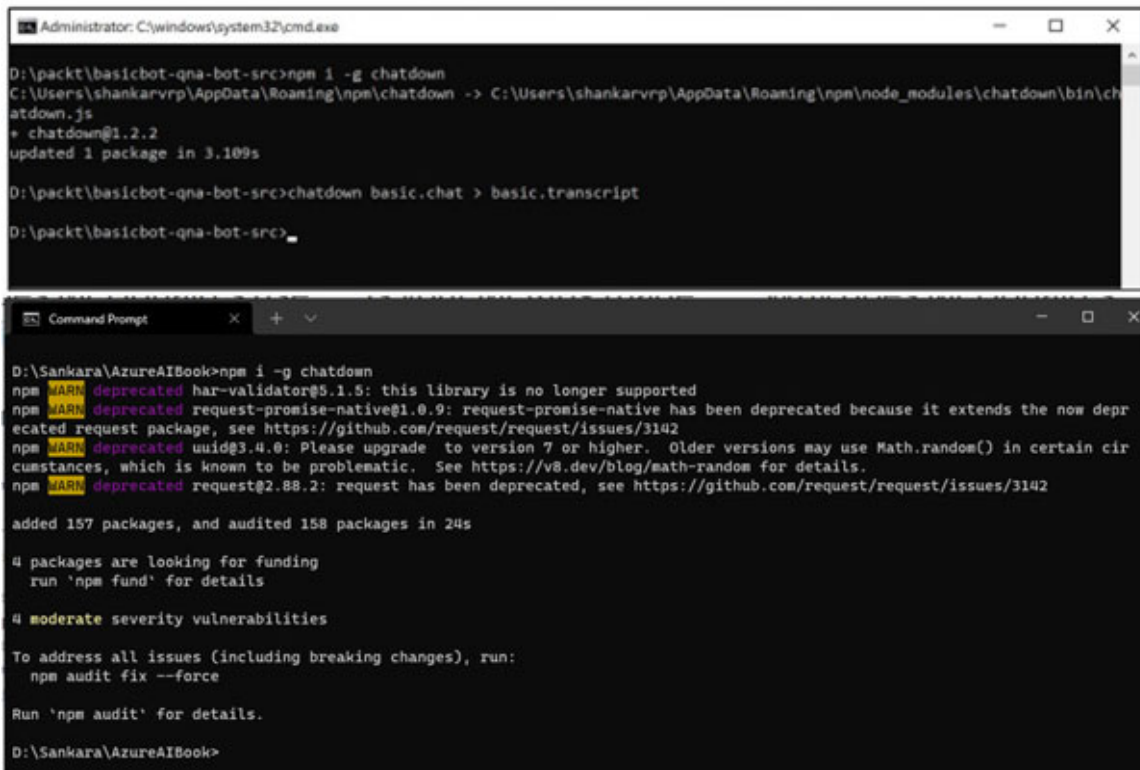
**Header** - defines the participants of the conversation

**Actual Conversation** - Back and forth conversation between a user and a bot

Chat files are created by a tool named “*Chatdown*” and it is one of the many tools available to work with development and testing of Bots. Chatdown can convert the `.chat` files to

`.transcript` files which then can be loaded on to the emulator and run to test a bot’s behavior and below is a quick test of this with the sample `.chat` file shown above.

Install Chatdown using `npm` and convert the `basic.chat` file.



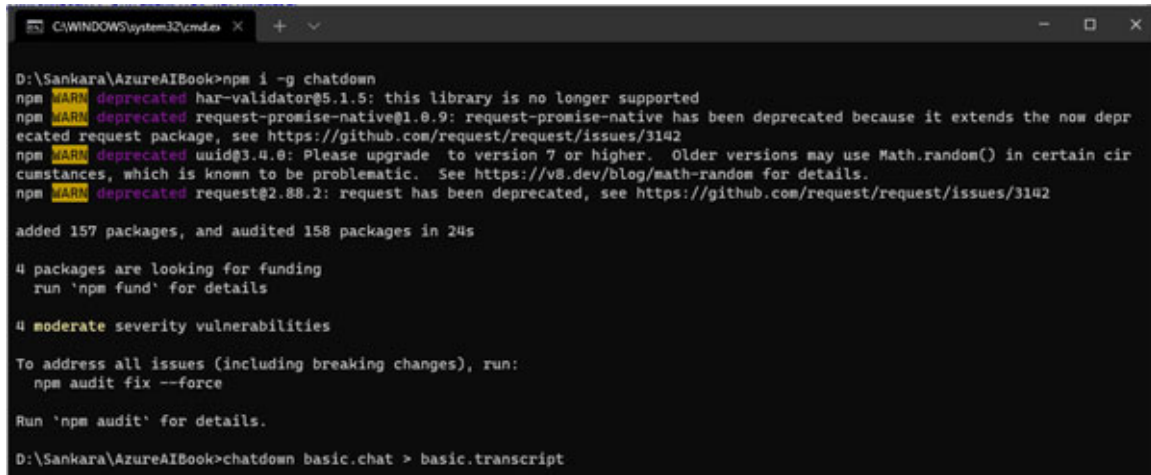
```
Administrator: C:\windows\system32\cmd.exe  
D:\packt\basicbot-qna-bot-src>npm i -g chatdown  
C:\Users\shankarvrp\AppData\Roaming\npm\chatdown -> C:\Users\shankarvrp\AppData\Roaming\npm\node_modules\chatdown\bin\chatdown.js  
+ chatdown@1.2.2  
updated 1 package in 3.109s  
  
D:\packt\basicbot-qna-bot-src>chatdown basic.chat > basic.transcript  
D:\packt\basicbot-qna-bot-src>  
  
Command Prompt  
D:\Sankara\AzureAIBook>npm i -g chatdown  
npm WARN deprecated har-validator@5.1.5: this library is no longer supported  
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated request package, see https://github.com/request/request/issues/3142  
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.  
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142  
  
added 157 packages, and audited 158 packages in 24s  
  
4 packages are looking for funding  
  run 'npm fund' for details  
  
4 moderate severity vulnerabilities  
  
To address all issues (including breaking changes), run:  
  npm audit fix --force  
  
Run 'npm audit' for details.  
D:\Sankara\AzureAIBook>
```

*Figure 6.26: Installing Chatdown*



.**transcript** files which then can be loaded on to the emulator and run to test a bot's behavior and below is a quick test of this with the sample .**chat** file shown above.

Install Chatdown using **npm** and convert the **basic.chat** file.



```
D:\Sankara\AzureAIBook>npm i -g chatdown
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated request package, see https://github.com/request/request/issues/3142
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 157 packages, and audited 158 packages in 24s

4 packages are looking for funding
  run 'npm fund' for details

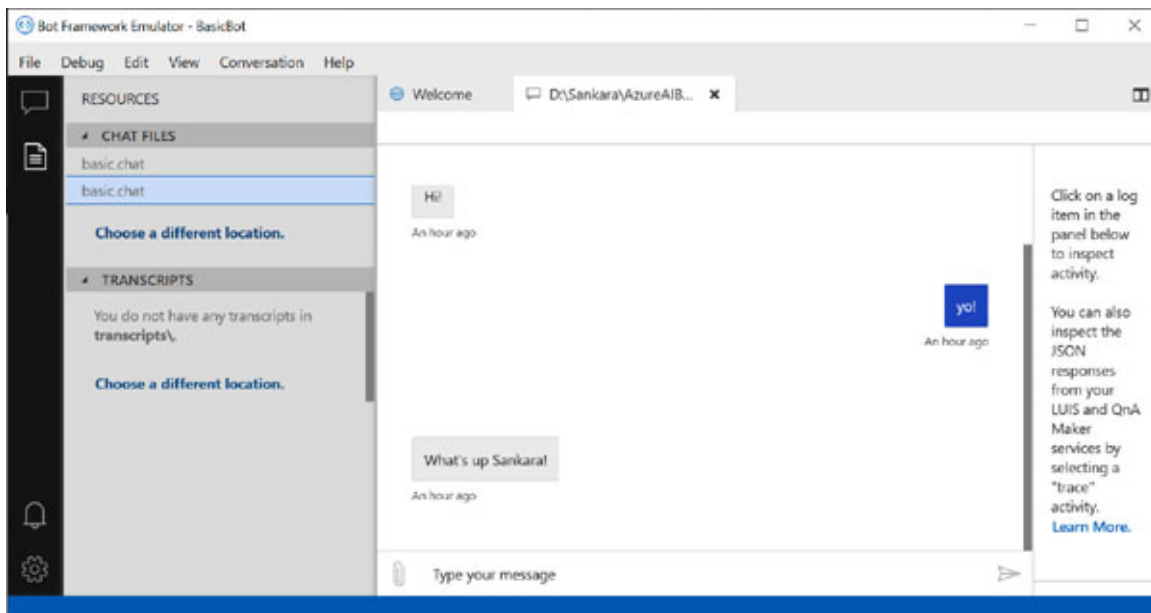
4 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

D:\Sankara\AzureAIBook>chatdown basic.chat > basic.transcript
```

*Figure 6.27: Convert **basic.chat** file*



*Figure 6.28: Loading a chat file directly*

Opening the transcript loads it up in the Emulator and runs the chat automatically, showing how it would have gone if the user was actually conversing.

Both Chat files and Transcript files can be accessed under the Resources section:



- **Logs:** Log section shown in the lower right area of the emulator records all the events and activities of the Bot which includes informational and error messages with respect to connectivity, trace etc. Take a look at the log section in [figure 6.20](#).
- **Inspector:** Using the INSPECTOR feature to the right of the emulator window, you can inspect the raw JSON activity of each message. When you click on the message bubble within the conversation window, the bubble will turn yellow and the JSON object activity will be shown on inspector window. Information in the JSON displayed, includes key metadata, including channelID, activity type, conversation id, the text message, endpoint URL, and so on.
- **Services:** Emulator has the option to add additional services such as LUIS, QnA, or dispatch models to it. Click on the (+) button on the Services pane when the bot is loaded and you would see options for adding LUIS, QnA Maker, and Dispatch.
- **Debugging:** One of the key features of the Emulator is that it allows to debug the interactions captured between a user and a bot. Clicking on any event/activity in the Log section shown in the lower right area of the emulator, information related to this specific interaction is displayed in the emulator's Inspector window in JSON format. This definitely helps you to understand every step of the conversation between the user and the bot's corresponding response, and debug to see if the response was as expected or if it failed or if it was a irrelevant and enables in resolving those issues.

Using the transcript files discussed earlier is one of the ways to record conversations and use them later for debugging interactions.

### [Adding LUIS to the sample](#)

In the real world applications of Bots, we would have to deal with different sets of entities and very many QnAs. Therefore, it is more likely that we will have to use multiple LUIS models and QnA Knowledge bases together to cover all business requirements and create a complete experience for users. To provide this experience, there should be logic to route the incoming user question to the appropriate intent of LUIS or the right knowledge base within QnA Maker KBs. This is where the dispatch model and the dispatch tool come in.

Dispatch tool evaluates LUIS models used to dispatch intent across multiple bot modules such as LUIS models, QnA knowledge bases and others. The tool manages this by creating a separate LUIS app to route user inputs to the correct model or QnA KB.

Dispatch model has to be used in the following scenarios as described in the Microsoft Documentation here:

1. Your bot consists of multiple modules and you need assistance in routing user's utterances to these modules and evaluate the bot integration.
2. Evaluate quality of intents classification of a single LUIS model.
3. Create a text classification model from text files.

With the sample “*Basic Bot*” that we developed in the earlier chapter, let us add natural language processing to it and also add a second knowledge base so that we can showcase how the bot is able to route requests with the help of dispatcher created routing model. The bot basically now becomes a “*dispatcher*”.

### [Adding a LUIS Model](#)

Create a LUIS Model by leveraging the pre-built “*Reservation*” domain models and for our purpose we will choose just a few of the reservation models available -

**RestaurantReservation.Confirm**

**RestaurantReservation.FindReservationEntry**

Name the LUIS app as “*BasicBotLuis*”.

### [Adding a second QnA KB](#)

We already created a KB as part of the “*Basic Bot*” we created in the previous section. Let us add a second knowledge base and this time we will choose a FAQ from the famous Italian restaurant - Olive Garden. The FAQ source we would use is from here.

You could refer to the previous chapter for a step by step on creating a KB in QnA.

### [Creating a dispatch model using the Bot Framework Orchestrator](#)

In the past, Microsoft Bot Framework used a dispatch tool for the “*Dispatching Model*” when multiple LUIS and QNA apps were used. Requests were dispatched to the right destination using this dispatch tool. This tool is now deprecated and replaced with the Bot Framework Orchestrator which comes as part of the Bot Framework CLI.

Orchestrator rather provides a straightforward solution for dispatching, it uses “*language understanding*” and at the same time simplifies the modelling process that eliminates deep expertise in neural networks or **natural language processing (NLP)** areas. Existing Bots that use Dispatch tool can also be migrated quite easily to switch to the Orchestrator model.

Let us get started with the Orchestrator tool to understand how requests are dispatched.

1. Start by installing the Bot Framework CLI with a **npm** command in the command line or PowerShell.

```
npm i -g @microsoft/botframework-cli
```

2. You can verify the installation of the CLI by using **bf** command. The version as of this writing is 4.15.0. Once created, you can start preparing the Orchestrator by adding LUIS application models and the QnA Maker knowledge bases as Orchestrator data sources.
3. Prepare the orchestrator by adding your LUIS Model and QnA KB.

- `bf orchestrator:add -t luis --id <luis-app-id> --endpoint <luis-authoring-endpoint> -k <luis-authoring-subscription-key> --routingName l_reservation`

The LUIS Authoring app is added to the orchestrator with the provided endpoint and application id. You can check the **orchestratorsettings.json** file at this point which is populated with just the app information.

4. Add the QnA Maker Knowledge Base as well to the orchestrator settings.

- `bf orchestrator:add -t qna --id <kb-id-of-mcdonaldsfaq-qna> -k <qna-subscription-key-of-mcdonaldsfaq-bot> --routingName q_mcdonaldsfaq`

Once the QNA Knowledge bases are added (you can verify the **orchestratorconfig.json**) the proceed with creating the language model for the orchestrator.

```

1  {
2    "modelPath": "D:\\Sankara\\AzureAIBook\\orchestrator-bot\\model",
3    "snapshotPath": "D:\\Sankara\\AzureAIBook\\orchestrator-bot\\generated\\orchestrator.blu",
4    "dataSources": {
5      "hierarchical": true,
6      "inputs": [
7        {
8          "type": "luis",
9          "id": "e51f5d1b-3610-472c-8c3d-b9c88006ada0",
10         "version": "0.1",
11         "key": "cbff18a8e73c48fca0e22a03d9d2d45c",
12         "endpoint": "https://reservationapp-authoring.cognitiveservices.azure.com/",
13         "routingName": "l_reservation",
14         "filePath": "D:\\Sankara\\AzureAIBook\\orchestrator-bot"
15       },
16       {
17         "type": "qna",
18         "id": "37112c08-a6f6-4d53-9df6-258692e9d73f",
19         "version": "",
20         "key": "205f474cd7d746c6a722aec16d9ab1bc",
21         "endpoint": "",
22         "routingName": "q_mcdonaldsfaq",
23         "filePath": "D:\\Sankara\\AzureAIBook\\orchestrator-bot"
24       }
25     ],
26     "path": "D:\\Sankara\\AzureAIBook\\orchestrator-bot\\CognitiveModels"
27   }
28 }

```

*Figure 6.29: Orchestrator Settings*

5. Download the base model and create a snapshot with the previously added LUIS and QNA KBs.
  - bf orchestrator:basemodel:get --out model
  - bf orchestrator:create --in CognitiveModels --model --out generated

```

PS D:\Sankara\AzureAIBook\orchestrator-bot> bf orchestrator:add -t luis --id e51f5d1b-3610-472c-8c3d-b9c8806ada0 --endpoint https
://reservationapp-authoring.cognitiveservices.azure.com/ -k cbff18a8e73c48fca0e22a03d9d2d45c -v 0.1 --routingName l_reservation
Added luis source with id e51f5d1b-3610-472c-8c3d-b9c8806ada0
PS D:\Sankara\AzureAIBook\orchestrator-bot> bf orchestrator:add -t qna --id 37112c08-a6f6-4d53-9df6-258692e9d73f -k 205f474cd7d746c
6a722a6c16d9ab1bc --routingName q_mcdonaldsfaq
Added qna source with id 37112c08-a6f6-4d53-9df6-258692e9d73f
PS D:\Sankara\AzureAIBook\orchestrator-bot> md model

Directory: D:\Sankara\AzureAIBook\orchestrator-bot

Mode                LastWriteTime         Length Name
----                -
d-----          21-02-2022    23:58         model

PS D:\Sankara\AzureAIBook\orchestrator-bot> bf orchestrator:basemodel:get --out model
Downloading model...
Total to download: 268112755 bytes...
OrchestratorBaseModel.getModelAsync(): model downloaded...
Model pretrained.20200924.microsoft.dte.00.06.en.onnx downloaded to model
PS D:\Sankara\AzureAIBook\orchestrator-bot> md generated

Directory: D:\Sankara\AzureAIBook\orchestrator-bot

Mode                LastWriteTime         Length Name
----                -
d-----          22-02-2022     00:00        generated

PS D:\Sankara\AzureAIBook\orchestrator-bot> bf orchestrator:create --in CognitiveModels --model model --out generated
Processing D:\Sankara\AzureAIBook\orchestrator-bot\CognitiveModels\dataSources\l_reservation.lu...
Processing D:\Sankara\AzureAIBook\orchestrator-bot\CognitiveModels\dataSources\q_mcdonaldsfaq.qna...
Snapshot written to D:\Sankara\AzureAIBook\orchestrator-bot\generated\orchestrator.blu
PS D:\Sankara\AzureAIBook\orchestrator-bot>

```

*Figure 6.30: Orchestrator Base Model*

You can see that this creates the corresponding files for LUIS and QnA and also writes a snapshot of the orchestrator to **orchestrator.blue**.

6. Check the model by running a sample query –

- `bf orchestrator:query -i="generated\orchestrator.blu" -m=model -q="special hours"`

```

PS D:\Sankara\AzureAIBook\orchestrator-bot> bf orchestrator:query -i="generated\orchestrator.blu" -m=model -q="frozen burger"
[
  {
    "label": {
      "name": "l_reservation",
      "label_type": 1,
      "span": {
        "offset": 0,
        "length": 13
      }
    },
    "score": 0.6711399913519643,
    "closest_text": "my restaurant reservation"
  },
  {
    "label": {
      "name": "q_mcdonaldsfaq",
      "label_type": 1,
      "span": {
        "offset": 0,
        "length": 13
      }
    },
    "score": 0.5755166026662271,
    "closest_text": "Are McDonald's burgers fresh or frozen?"
  }
]

```

*Figure 6.31: Verify Base Model*

7. In the downloaded code, modify the `appsettings.json` file to include details of LUIS app and the QnA KB.

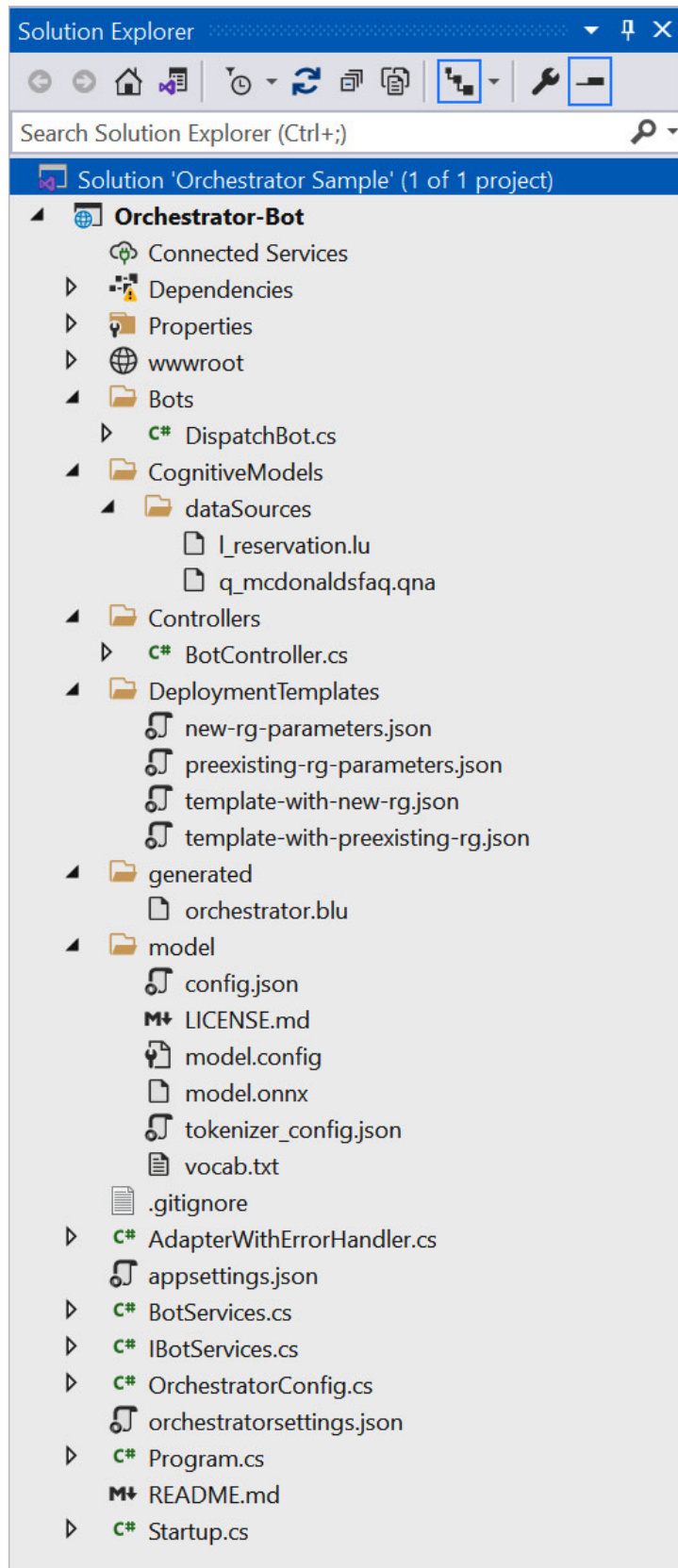
```
1  {
2    "MicrosoftAppType": "",
3    "MicrosoftAppId": "",
4    "MicrosoftAppPassword": "",
5    "MicrosoftAppTenantId": "",
6
7    "QnAKnowledgebaseId": "37112c08-a6f6-4d53-9df6-258692e9d73f",
8    "QnAEndpointKey": "a8daecb5-05d8-4804-a8ed-3a25a42e2a6e",
9    "QnAEndpointHostName": "https://qna-mcdonalds.azurewebsites.net/qnamaker",
10
11    "LuisReservationAppId": "e51f5d1b-3610-472c-8c3d-b9c88006ada0",
12    "LuisAPIKey": "1b73143a34bf46afa79d8fb23152ae9f",
13    "LuisAPIHostName": "https://reservationapp.cognitiveservices.azure.com/",
14
15    "AllowedHosts": "*",
16
17    "Orchestrator": {
18      "ModelFolder": ".\\model",
19      "SnapshotFile": ".\\generated\\orchestrator.blu"
20    }
21  }
```

*Figure 6.32: appSettings.json should be updated*

Notice the 3 folders that have been created during the process.

- **model folder:** Has the orchestrator's base model files.
- **CognitiveModels folder:** Holds the generated files for the respective applications – LUIS , QnA.
- **generated folder:** Contains the orchestrator snapshot that includes all intents, utterances and their scores.

Here is a view of the solution explorer that shows the entire code base with the most important ones expanded.





*Figure 6.33: Solution Explorer*

## Updating the BasicBot source code to include dispatch support

Now that we have the models up and running, we have to modify the code add the dispatcher functionality into the bot and also add LUIS and QnA KBs to the bot so that it can reach out to those services based on routing of incoming requests.

### **IBotService.cs and BotServices.cs**

BotServices is the class that defines the Bot services connection to the dispatch model and other LUIS models and KBs. It implements an interface **IBotService** that defines the connection points.

```
public class BotServices : IBotServices
{
    public BotServices(IConfiguration configuration,
        OrchestratorRecognizer dispatcher)
    {
        // Read the setting for cognitive services (LUIS, QnA) from
        // the appsettings.json
        // If includeApiResults is set to true, the full response
        // from the LUIS api (LuisResult)
        // will be made available in the properties collection of the
        RecognizerResult LuisReservationRecognizer =
        CreateLuisRecognizer(configuration, "LuisReservationRecognizer
        ");
        Dispatch = dispatcher;
        McDonaldsQnA = new QnAMaker(new QnAMakerEndpoint
        {
            KnowledgeBaseId = configuration["QnAKnowledgebaseId"],
            EndpointKey = configuration["QnAEndpointKey"],
            Host = configuration["QnAEndpointHostName"]
        });
    }
    public OrchestratorRecognizer Dispatch { get; private set; }
    public QnAMaker McDonaldsQnA { get; private set; }
    public LuisRecognizer LuisReservationRecognizer { get; private set; }
    private LuisRecognizer CreateLuisRecognizer(IConfiguration
        configuration, string appIdKey)
```



```

{
    var luisApplication = new LuisApplication(
        configuration[appIdKey],
        configuration["LuisAPIKey"],
        configuration["LuisAPIHostName"]);
    // Set the recognizer options depending on which endpoint
    // version you want to use.
    // More details can be found in
    // https://docs.microsoft.com/en-gb/azure/cognitive-
    // services/luis/luis-migration-api-v3
    var recognizerOptions = new LuisRecognizerOptionsV2(luisApplication)
    {
        IncludeAPIResults = true,
        PredictionOptions = new LuisPredictionOptions()
        {
            IncludeAllIntents = true,
            IncludeInstanceData = true
        }
    };
    return new LuisRecognizer(recognizerOptions);
}
}

```

## ErrorHandlerAdaptor

Error handler to handle errors encountered by the Bot, where custom error messages can be defined for users to see rather than error messages thrown at them.

```

public class AdapterWithErrorHandler : CloudAdapter
{
    public AdapterWithErrorHandler(BotFrameworkAuthentication auth,
        ILogger<IBotFrameworkHttpAdapter> logger, ConversationState
        conversationState = null)
        : base(auth, logger)
    {
        OnTurnError = async (turnContext, exception) =>
        {
            // Log any leaked exception from the application.

```

```

logger.LogError(exception, $"[OnTurnError] unhandled error :
{exception.Message}");
// Send a message to the user
await turnContext.SendActivityAsync("The bot encountered an
error or bug.");
await turnContext.SendActivityAsync("To run this sample make
sure you have the LUIS and QnA models deployed.");
await turnContext.SendActivityAsync("To continue to run this
bot, please fix the bot source code.");
if (conversationState != null)
{
    try
    {
        // Delete the conversationState for the current
        conversation to prevent the
        // bot from getting stuck in a error-loop caused by being
        in a bad state.
        // ConversationState should be thought of as similar to
        "cookie-state" in a Web pages.
        await conversationState.DeleteAsync(turnContext);
    }
    catch (Exception e)
    {
        logger.LogError(e, $"Exception caught on attempting to
        Delete ConversationState : {e.Message}");
    }
}
// Send a trace activity, which will be displayed in the Bot
Framework Emulator
await turnContext.TraceActivityAsync("OnTurnError Trace",
exception.Message,
"https://www.botframework.com/schemas/error", "TurnError");
};
}
}

```

## **Build and Test it using Emulator**

Build the code using Visual Studio and run it. By default the bot is hosted on localhost on port 3978 and visual studio opens up a browser with the default bot home page with some instructions on how to connect to the bot.

Let us check how the bot in the emulator. Start the emulator and connect to the running bot by connecting to its endpoint url – **http://localhost:3978/api/messages**.

You should see a welcome message that says “*Welcome to Restaurant bot User. You can search and reserve a seat at McDonalds*”. And when you go ahead and ask a question “*do you serve frozen burgers?*” or even just send a phrase such as “*frozen burgers*”, the bot rightly answers the question. However, when you ask a common questions such as “*where to find McDonalds?*” or ask “*Do i have a table reserved at McDonalds this evening?*” the code is written in a way to respond back with the intent determined by the dispatcher bot. Bot responds with the “**RestaurantReservatoin.FindReservationEntry**” as the intent. When you follow-up by “*can you confirm a reservation for me tonight?*”, the bot rightly picks up the “**RestaurantReservation.Confirm**” entity denoting that the question’s intent was to confirm the restaurant. The bot code is basic and right now only written to display the chosen intent and the related intents to user.

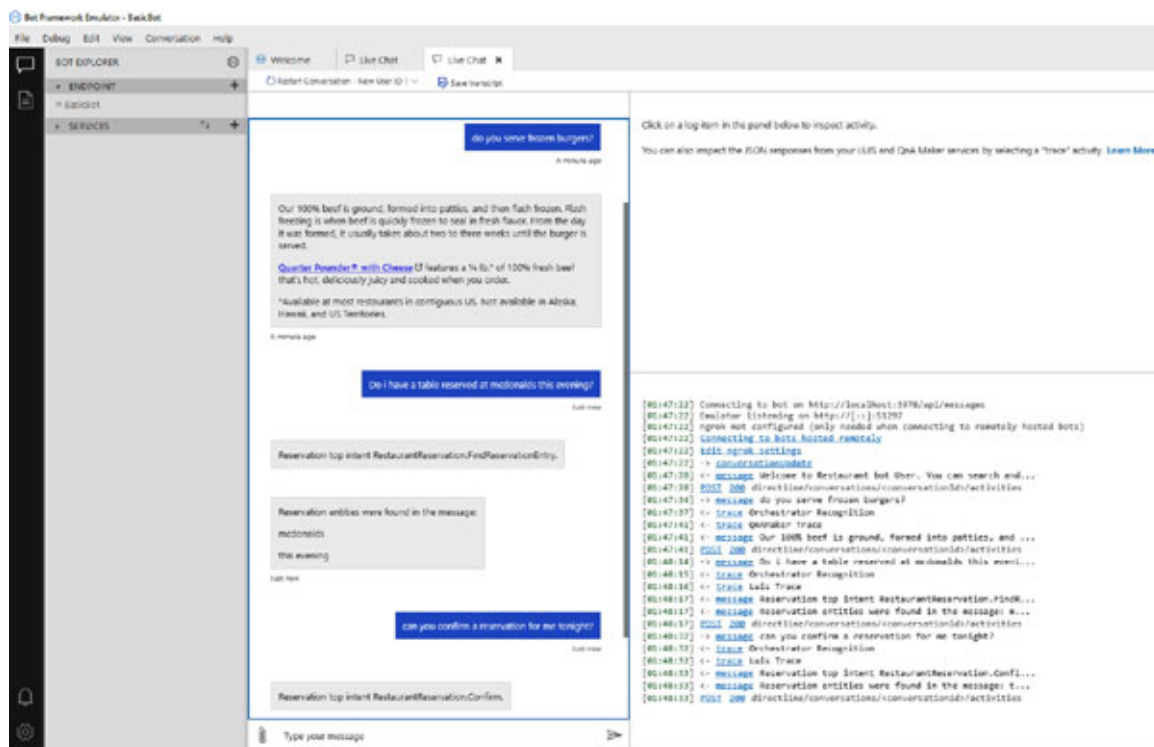


Figure 6.34: Build the code and Test it in Emulator

## [Adding the Oliver Garden FAQ as an exercise](#)

Now as an exercise, you could take the other QnA KB you created – the Olive Garden FAQ and incorporate it into this project and in the orchestrator and try and test the same.

A similar command to the one below can be used to add Olive Garden FAQ

- `bf orchestrator:add -t qna --id <kb-id-of-olivegardebfaq-qna> -k <qna-subscription-key-of-olivegardenfaq-bot> --routingName q_olivegardenfaq`

## [Enhancing it further](#)

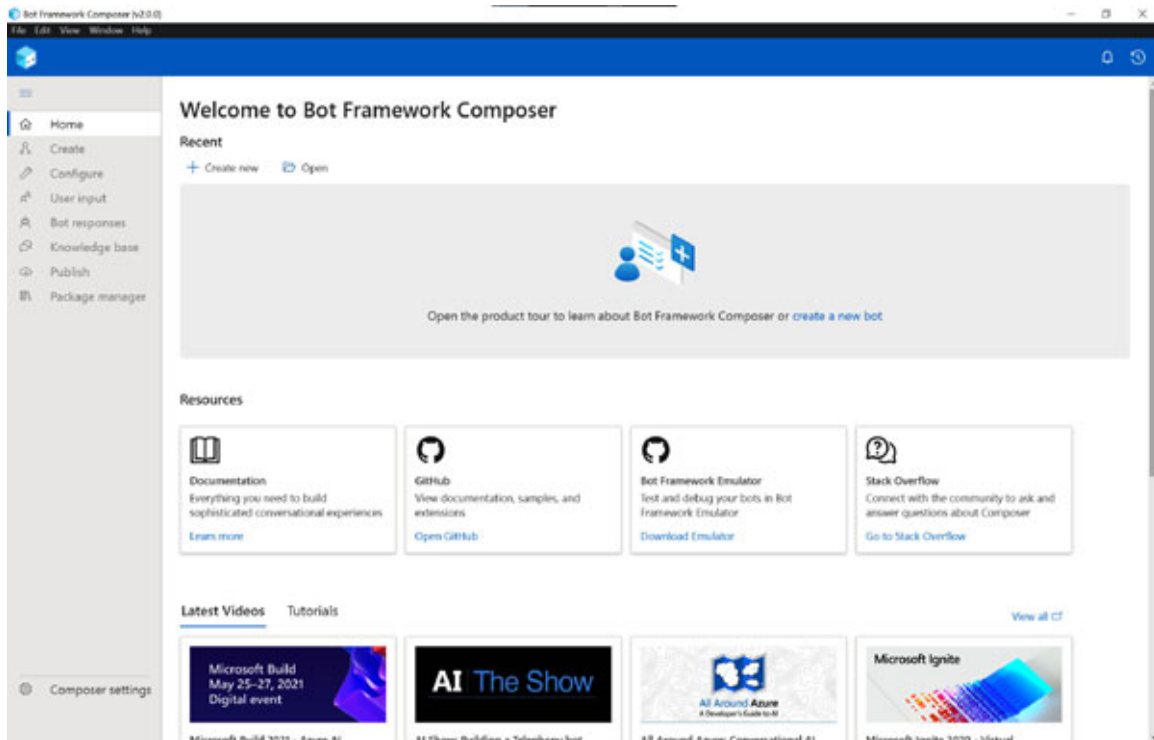
This sample doesn't show any dialogs, but you could enhance the conversation by using dialogs as response to the intents and accordingly have multiple sub dialogs as well based on the now the conversation goes.

## [Building the Bot Framework Composer](#)

Composer provides a visual interface represented by dialogs to build bots. One would choose composer over Bot Framework, if they intend to write less code and use built-in building blocks. The current version of Composer is 2 and fairly mature.

We will not go through the experience of creating a new bot via the Composer since the end product is a Bot which is similar to the one you would create using the Bot Framework SDK. I would like to just show the visual interface and how it looks like for a bot that is built and complete.

The home page of the composer is quite like a dashboard that shows recently created bots via composer and resources for learning.



*Figure 6.35: Bot Framework Composer*

**Note: You need Azure Functions Core Tools to be installed, you would be prompted to install it before you can proceed if it's not already installed.**

You can follow this Microsoft tutorial to use Bot Composer and create a weather bot - Tutorial: Create a weather bot with Bot Framework Composer | Microsoft Docs

## Enterprise Scenarios

In one of the Microsoft's Build developer conferences, Microsoft's CEO Satya Nadella shared his vision of the future that involved chatbots, machine learning, and artificial intelligence and he also talked about how "*bots are the new apps*" that advance the human machine interaction.

The rapid pace of change in today's "*transformative*" world makes it important for enterprises to learn quickly, and continuously iterate in an agile way. Bots getting into enterprise scenarios, has closed in the distance between any enterprise software and its users by humanizing and personalizing content and empowering them with information in the digital world.

## A Customer scenario

A customer, John Doe, is about to make an important purchase – his dream car and contacts his bank to get updated on his financial position. In a typical interaction with a bank, that would likely be the end of the story, but what if the bank can step beyond transactional enquiries and be a financial advisor?

With a Bot, John can start with asking: “*How much money do I have?*”. That triggers a conversation flow with John to help him with his inquiry. The bot understands the intent and provides the type of assistance he is looking for. It uses LUIS to determine how to handle this inquiry. If the question was “*How much money do I have in my current account?*”, it could still detect the account and return the balance.

However, the bot also provides a summary of his spending in rich content that is embedded for instance, in a chat window, which allows the flow of conversation such as “*Show me my top expenses in the last 6 months*” or “*How much have I spent on travel this year?*” BOTs can go on and also show credit scores, remind credit card payments, all of which provides John with more insights to make his decision about buying a car.

Fast forward, and if John decides to buy the car, he needs an insurance. Insurance companies can leverage business analytics and AI to fundamentally transform the way they do customer service and reduce customer churn. John goes to an insurance provider’s website for an insurance quote. A Bot verifies his information and asks for the details of the car including its picture and in parallel, Bot accesses John’s profile in Dynamics CRM and recognizes that the customer has a good history of payments and decides to proceed with the processing of his application.

This shows that Bots can enable different interactions patterns with customers, and they represent the opportunity to empower them to interact with businesses in natural and personal ways. They have the ability to identify the intent of the question regardless of how it’s phrased by the user. We can type these questions or even leverage speech-to-text for an even more natural pattern of interaction. The next interaction flow is more complex.

## Typical use cases

The journey of digital business transformation can begin anywhere in an organization and at a minimum, there could be the following key areas for transformation:

## **Re-Interpret customer interactions**

It all starts with customers. Enterprises are continuously, aggressively exploring opportunities to add more value to customers. The use of digital technologies to provide connected products and services that engage customers with personalized experiences, provide the recommended offers and value-added services at the right time.

“*Chat bots*” are the new user interface paradigm, that exposes software services through conversational interfaces, what is termed as “*Conversational AI*”. “*Conversational AI*” is a subfield of AI that produces seamless conversations between humans and computers.

## **Re-Engineer operational processes**

Companies have started infusing digital into their operations, and it has become easier than ever to grow through the utilization of digital tools and provides immense opportunities to optimize operations. From physical plants where assets can be maintained more predictably and proactively, to more adaptable and personalized business processes, to also influence their innovation process.

This is where “*Automation Bots*” observe the user behaviour and automates tasks combining information from varied data sources and presents with insights for the businesses and its customers.

## **Re-Imagine enterprise models**

Enterprises have started using digital technologies to completely re-imagine business models. While some target delivering new digital products with the data available in abundance, others are complementing the physical experiences with re-defined digital ones. Such momentum has driven enterprises to re-strategize their thinking and move towards “*digital services*” rather than just focusing on “*building products*”.

“*Digital Assistants*” packed into electronic devices provide personalized and customized experienced targeted at individuals rather than a standard common solution approach.

Multiple different enterprise verticals use bots across the board such as

Telecommunication, automotive, healthcare, education, insurance, government, finance. Vodafone’s – Tobi is a good recent example of how customer engagement has been redefined via conversational interfaces.

## **Leveraging Natural Language Processing**

Conversational chatbots were made to assist organizations with customer correspondences, answer questions and take orders. To play out these errands effectively, they should be insightful. At the end of the day, great at **Natural Language Processing (NLP)**. Be that as it may, the point is, it's pretty tedious to fabricate your own NLP. Fortunately, you don't generally need to begin starting with no outside help. Natural Language Processing engines have become better than ever in the recent years. Coupled with deep learning and neural networks, they are now empowering us to build highly engaging and human-like interfaces.

## **Conclusion**

We introduced you to the Bot Framework and the basic constructs of how Bots could be developed, debugged, and deployed with ease on Azure with Microsoft provided SDKs and Tools. This should help you get started on the framework and let you explore more advanced scenarios such as using dialogs, integrating different channels, connecting to various other cognitive intelligence APIs to build innovative solutions for customers and enterprises. In the next chapter we will learn about ML.NET. We will also understand how it can be invoked from .NET code.

With that, we will look at some industry scenarios and how AI gets integrated in the following Appendix section.

## **Thought Experiment**

### **1. How do you test and debug your bots?**

- Use Visual Studio and use the Emulator integrated within Visual Studio IDE
- Use VS Code and use the Debug option within VS Code
- Use the Bot Emulator outside of any IDE to test and debug bots
- Publish in Azure and debug

### **2. Can you run bots in your local machines?**

- They must be published to Azure to work, they don't work locally
- They can be used locally with Bot Emulator



**3. What debugging options does the Bot framework and the emulator provide?**

- Local Debugging with emulator
- Remote Debugging with emulator
- Both

**4. What are the cognitive skills that bots can be enabled with?**

- LUIS
- QnA Maker
- Azure Search
- Speech Service
- Vision/ Custom Vision Service
- Bing Search
- Bing Spell Check
- All of the above

# CHAPTER 7

## Infusing ML in Custom Applications

### Using ML.NET

In the last few chapters, we have read about vision API and how we can leverage cognitive services to infuse intelligence into applications. In this chapter we will learn about ML.NET which is an easy to use .NET Machine Learning platform which helps in infusing intelligence into applications. ML.NET is a cross-platform, open source machine learning framework which helps in developing simple intelligent .NET applications.

The framework is built on .NET Core and .NET Standard. It supports multiple platforms which include Mac, Linux and Windows. The origin of ML.NET is from **text mining search and navigation (TMSN)** which was used internally in Microsoft products. The ML.NET repository consists of .NET APIs for consumption and training models. It helps in pre-processing, feature engineering, modeling, evaluation, and invoking.

With ML.NET, we do not need to learn a new programming language, like Python develop our own ML models. The framework provides capabilities which help us to train build, train, and deploy custom machine learning models locally without requiring any machine learning experience. After reading this chapter we will learn basics of ML.NET and how to leverage ML.NET.

### Structure

In this chapter, the following topics will be covered:

- Introducing ML.NET – how to leverage ML.NET to develop models
- Lab - Developing a sample ML Model
- Exposing the ML model as a service and invoking the service from .NET Core client

## Objective

After reading this chapter, we will understand how a machine learning model can be developed using ML.NET and then exposed as an API or a web service. We will also understand how it can be invoked from .NET code.

In this chapter, we will understand the basics of ML.NET:

- ML.NET – An introduction learn how to leverage ML.NET to develop models
- Lab -Developing a sample ML Model
- Exposing the ML model as a service and invoking the service from .NET Core client
- A thought experiment, which is a set of questions designed to aid understanding of the key concepts covered in this chapter

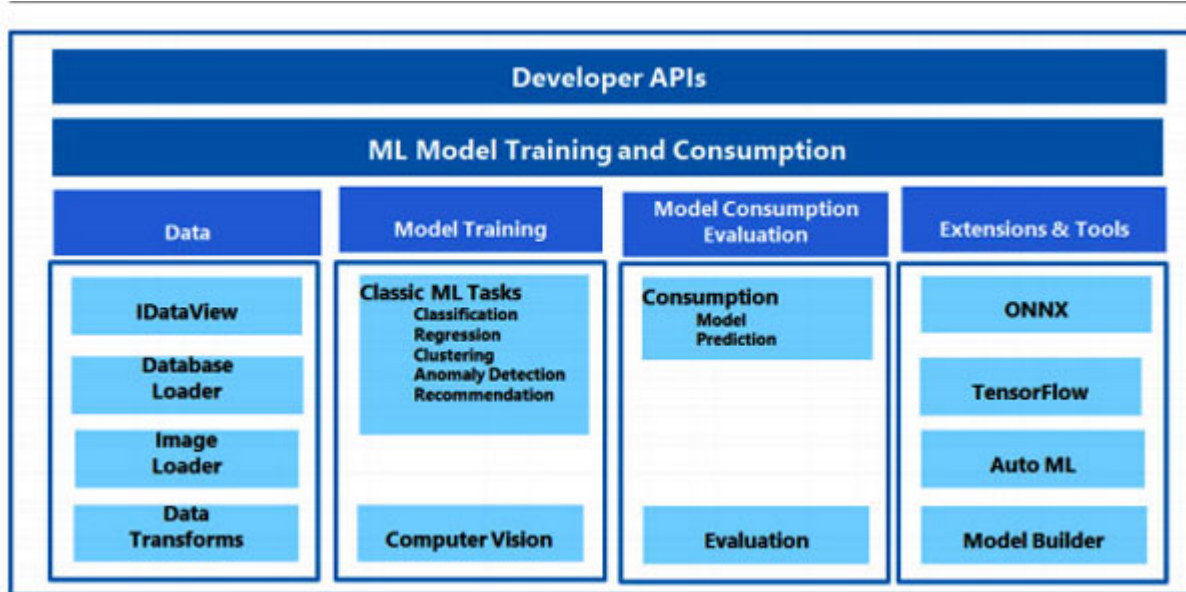
After the chapter, we will get a fair idea of ML.NET, the different algorithms supported, developing models and on exposing models as a service.

## Introducing ML.NET

ML.NET is a cross-platform, open source machine learning framework to easily add machine learning to simple .NET applications. ML.NET was initially developed in Microsoft Research. It is used in a number of Microsoft products which includes Windows, Bing, Azure etc. ML.NET helps perform complex Machine Learning tasks like regression, classification etc. easily and invoke the same through .NET code.

It has a set of learning algorithms, core ML data structures and transforms. It exposes .NET APIs for training the models and making predictions using the models. ML.NET comes with the support for the types and run time needed for all aspects of machine learning which includes core data types, extensible pipelines, high performance math, and data structures for heterogeneous data, tooling support, and more.

Following figure captures the key components of ML.NET framework:



*Figure 7.1: ML.NET Framework*

The following is a brief overview:

**Developer Friendly APIs:** Training APIs: These APIs help in creating and building models. Consumption APIs: These APIs help in making predictions

## Data:

- **IDataView:** IDataView is the data pipeline machinery. It consists of a set of interfaces and classes which help in transformation of data. It helps in handling high dimensional data and large data sets. IDataView related components include loaders, transforms, savers, trainers, predictors, etc. We can load datasets into an IDataView from any data source. We can use File Loaders for typical sources which include text, binary, and image files, and we can use the Database Loader to load and train data directly from relational databases like Oracle, SQL Server, and so on.
- **Data Transforms:** ML.NET provides a number of data transforms to convert data to an acceptable format for the ML algorithms like text featurizers.
- **Model Training:** Classic ML Tasks: ML.NET supports many logarithms like classification, regression, clustering, anomaly

detection, recommendation engine etc. It provides more than 40 trainers to select for a specific task. 8. Extending leveraging ML.NET [Chapter 1, 'Azure AI Platform and Services' \[ 4 \]](#) Computer Vision: ML.NET also offers TensorFlow based image classification/recognition with our own custom images.

- **Model Consumption and Evaluation:**

**Model Consumption:** Once the model is trained we can use the prediction engine to predict.

**Model Evaluation:** We can determine the accuracy of the model using evaluators provided by ML.NET.

- **Extensions:** ML.NET helps simplify other Machine Learning infrastructure libraries and runtimes, such as Cognitive ToolKit, TensorFlow and ONNX. The Microsoft **Cognitive Toolkit (CNTK)** is a tool kit which is open source and provides deep learning capabilities. TensorFlow is a library to accelerate machine learning and deep neural network research which is developed by the Google Brain Team. ONNX is Open Neural Network Exchange format. It is an open source shared model representation which helps in framework interoperability and shared optimization. ML.NET also offers Python bindings called NimbusML. We can create ML.NET models in Python with NimbusML. These models can be consumed in .NET applications.
- **Tools:** We can use ML.NET's tools (Model Builder in Visual Studio or the cross platform CLI) to make model training even easier. These tools use the ML.NET AutoML API internally to try many different combinations of algorithms and configurations in order to find the best model for our data and scenario. Now we will discuss the algorithms that are supported by ML.NET.

## [Algorithms supported by ML.NET](#)

ML.NET supports a number of training algorithms. An algorithm is a math that executes to define the model that will be produced with unique characteristics. With ML.NET, the same algorithm can be applied to different kinds of tasks while how the output of the algorithm is interpreted

differs. For instance, Stochastic Dual Coordinated Ascent can be used for Regression and Binary Classification.

## Choosing the right algorithm

Machine learning algorithms can be classified as supervised (Classification, regression) and unsupervised (clustering).

- **Supervised:** Supervised learning is the learning of the model with labelled input data and an algorithm to map the input data to the output data. The algorithm learns by predicting the output from input data and can predict the output where a new input value is provided. It involves offline analysis of data. Accuracy is higher.

Supervised learning can be further classified into:

- **Classification:** This type of learning is applicable to predict an output variable where the output is a category like the color of a dress etc.
- **Regression:** This type of learning is applicable to predict an output variable continuous value like predicting taxi fare for a given trip based on certain conditions.
- **Unsupervised:** This type of learning learns from unlabeled input data where there is no corresponding output variable. It involves real time analysis of data.
- **Clustering:** A clustering problem is where we want to discover grouping in the input data provided like grouping customers by segments. The model learns by looking at training data that are similar to each other and clusters them together like similar kind of birds.
- **Association:** An association rule learns by looking at the correlation of features in existing sample data. For instance, suggesting accessories for a dress purchased.
- **Semi-supervised:** This type of learning is applicable where some portion of the input data is labeled but most portion of it is unlabeled.

A number of factors determine the right algorithm to choose. The choice of the right algorithm depends on the problem we are solving, the characteristics of input data, and the compute and storage resources.

Sometimes we might have to try different algorithms to find the one that works best for the problem statement.

Algorithms work on features which are numerical values that are computed from input data. We can convert raw data into features using one or more data transforms. Transformed features are referred to as featurized text or featurized image data, etc.

Following is a view of the algorithms to be used for each type of task:

	Regression	Two Class Classification	Multi Class Classification
Average Perceptron			
Decision Forest			
Decision Jungle			
Decision Tree			
Fast Forest			
Linear Regression			
Bayes Linear Regression			
Log Regression			
Neural Network			
Ordinal Regression			
Poisson Regression			
SVM			
SVM Deep Support			

*Figure 7.2: Algorithms supporting various ML tasks*

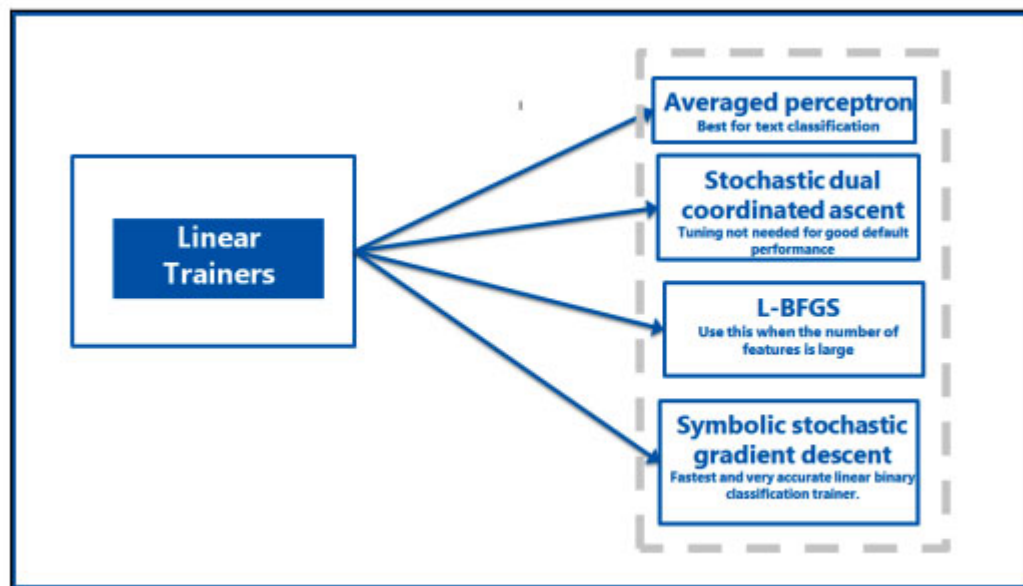
For each algorithm/task combination, ML.NET provides a trainer that executes the training algorithm. We will now discuss trainers.

## Trainers

For each combination of an algorithm and a task, ML.NET provides a trainer component that executes the training algorithm and performs the interpretation. For instance, for regression the **SdcaRegressionTrainer** leverages the **StochasticDualCoordinatedAscent** algorithm.

Given below are the some of the Trainers supported by ML.NET:

- **Linear Trainers:** Linear trainers create a model that calculate scores from a linear combination of the input data and a set of corresponding weights. Linear algorithms work well for features that are linearly separable.



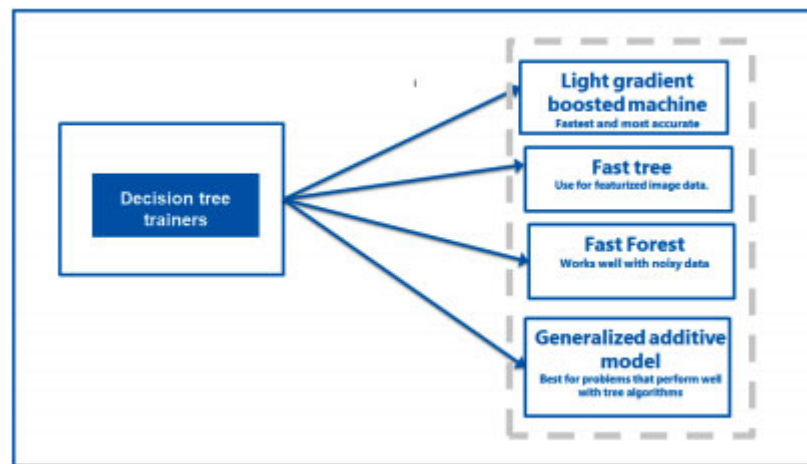
*Figure 7.3: Linear Trainers*

- **Averaged Perceptron:** This is used to classify inputs into several possible outputs based on a linear function, and then combined with a set of weights that are derived from the feature vector. This is good for text classification.
- **Stochastic dual coordinated ascent:** This is used to solve regularized loss minimization problems in machine learning.
- **L-BFGS:** This is an optimization algorithm used for parameter estimation to train various machine learning models using a limited amount of memory. This is used when the number of features is huge.



- **Symbolic Stochastic Gradient Descent:** This is used for minimizing loss functions with the form of a sum. This is very accurate and very fast.
- **Decision Tree Trainers:** Decision tree trainers are applicable to features that are not separable linearly. These algorithms create a model which is based on a decision series. The features for a decision tree algorithm need not be normalized and are very accurate.

Following are the various decision tree trainers in ML.NET:



*Figure 7.4: Decision Tree Trainers*

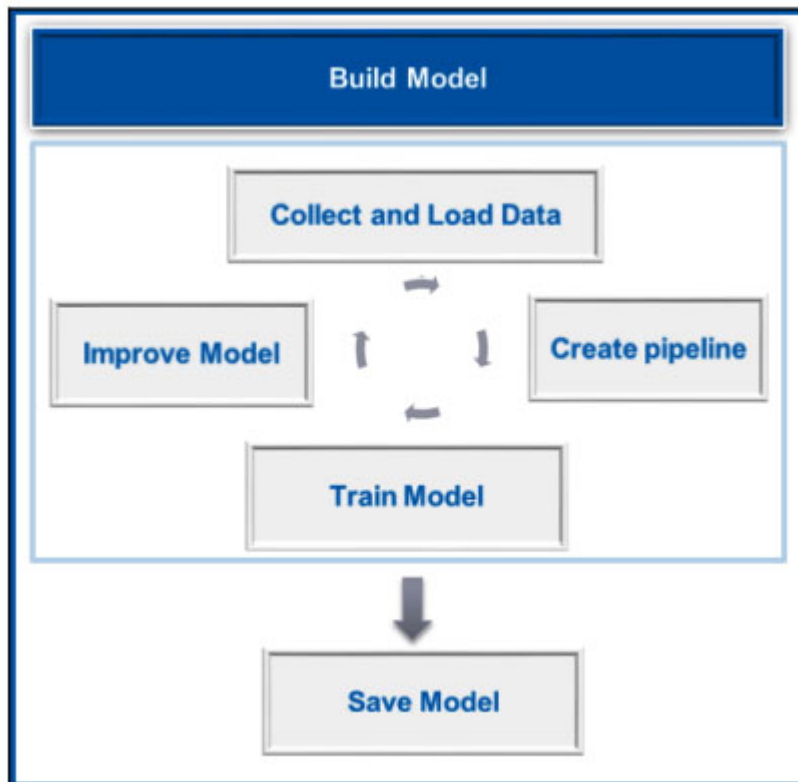
- **Light Gradient Boosted Machine:** This is a gradient boosting framework and is capable of handling large data and is designed to be distributed and efficient with faster training speed, lower memory usage and higher accuracy and efficiency.
- **Fast Tree:** Fast Tree does not require the entire data set to be read for learning hence reducing the time for learning and is efficient in terms of memory as well. Fast tree performs a test and train process each time test data arrives. This works well when the number of features is very high.
- **Generalized Additive Model:** This is used when a linear predictor depends linearly on unknown smooth functions.

Boosted decision trees are an ensemble of smaller trees. Each tree in the ensemble scores the input data. This input data is passed to the next tree to

produce a better score. Now we will look at the steps involved in creating a ML.NET model.

## Creating a ML.NET Model

A Machine Learning model helps in performing the transformation on input data and gives as output the predicted data. The basic steps in creating a model are given as follows:



*Figure 7.5: Steps in creating ML.NET Model*

**Step 1:** Collect Data and Load Data: ML.NET uses classes to define input data models ML.NET supports loading data from data sources that include data files In-memory collections JSON/XML Databases.

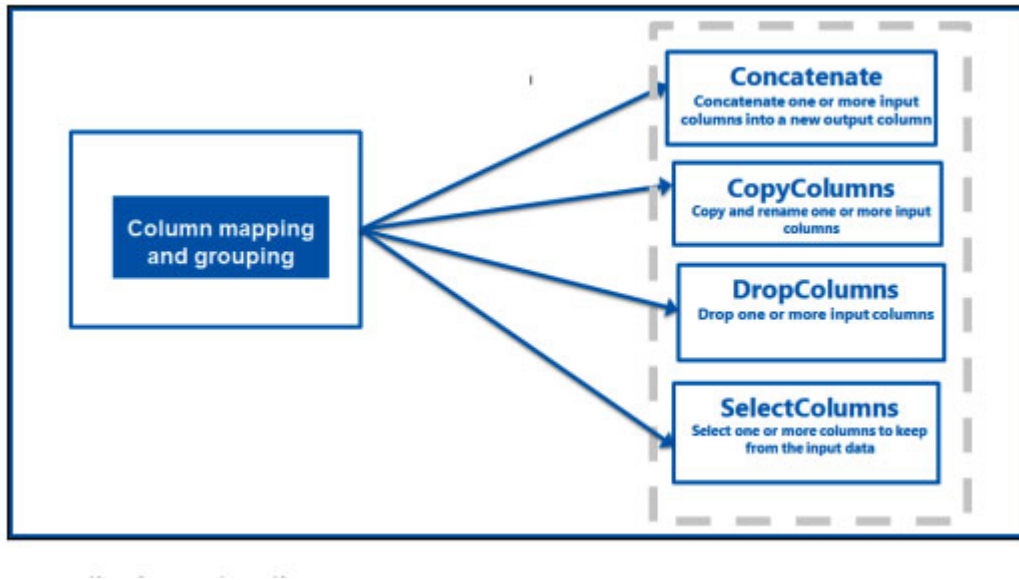
**Step 2:** Create Pipeline

Once the data is loaded, we need to prepare the input data. Input data is not clean and some data will be missing. We need to clean the data to be in a format expected by ML.NET algorithms.

- **Filtering:** We can use the filter options to remove unconnected data by using filtering.
- **Missing values:** We can deal with missing values by replacing them with a value like the mean value or some default value.
- **Normalizers:** We can use normalization to standardize features which are not on the same scale.
- **Min-Max normalization:** We can use this to normalize values based on the observed minimum and maximum values of the data.
- For example, we can normalize original income values [100000, 200000] to [0.5,1] which will give an output value in the range of 0-1.
- **Binning:** We can use binning to convert continuous values like age into a discrete representation of the input like age brackets of 0 to 15 etc.
- **Categorical Data:** We can use `OneHotEncoding` method to convert Non-numeric categorical data to a number.
- **Text Data:** We can use `FeaturizeText` to convert text to a numerical vector.

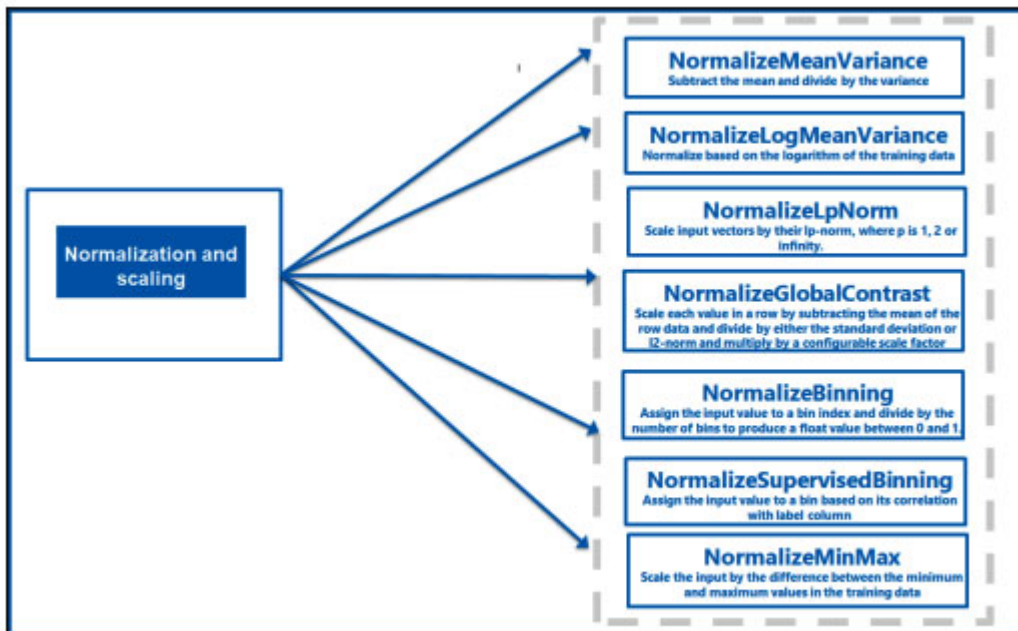
We can use **Data transformations** to prepare data for model training. Data transformations can be chained with each transformation in the chain expecting and producing data of specific types and formats.

**Column mapping and grouping:** Given below are options for column mapping and grouping:



*Figure 7.6: Column Mapping and Grouping*

**Normalization and Scaling:** Given below are options for normalization and scaling.



*Figure 7.7: Normalization and Scaling*

- **Conversions between data types:** Includes functions to convert the type of an input column to a new type, Map values to keys, map values to value, map values to vector, hash, and more.

- **Text transformations:** Includes functions to transform a text column into a float array of normalized ngrams and char-grams counts, split one or more text columns into individual words, and so on. Image transformations help convert an image to grayscale, convert a vector of pixels to ImageDataViewType, and so on.
- Categorical data transformations help convert one or more text columns into one-hotencoded vectors and convert one more text columns into hash-based one-hot encoded vectors.
- Missing values help create a new Boolean output column, the value of which is true when the value in the input column is missing and create a new output column, the value of which is set to a default value if the value is missing from the input column, and the input value otherwise.

### Step 3: Train Model

We have to separate data into two sets: training and testing. A large portion is used for training while a small portion is used for testing. We can minimize the effects of data discrepancies by using similar test and training data.

Cross-validation is a training and model evaluation technique which splits the data into several partitions and trains multiple algorithms on these partition to improve the robustness of the training process. This is an effective method for training when the datasets are small.

**Step 4: Improve Model** After training we test the model whether it is making the right predictions. We have to use the Evaluate method which produces different metrics depending on the task that is performed.

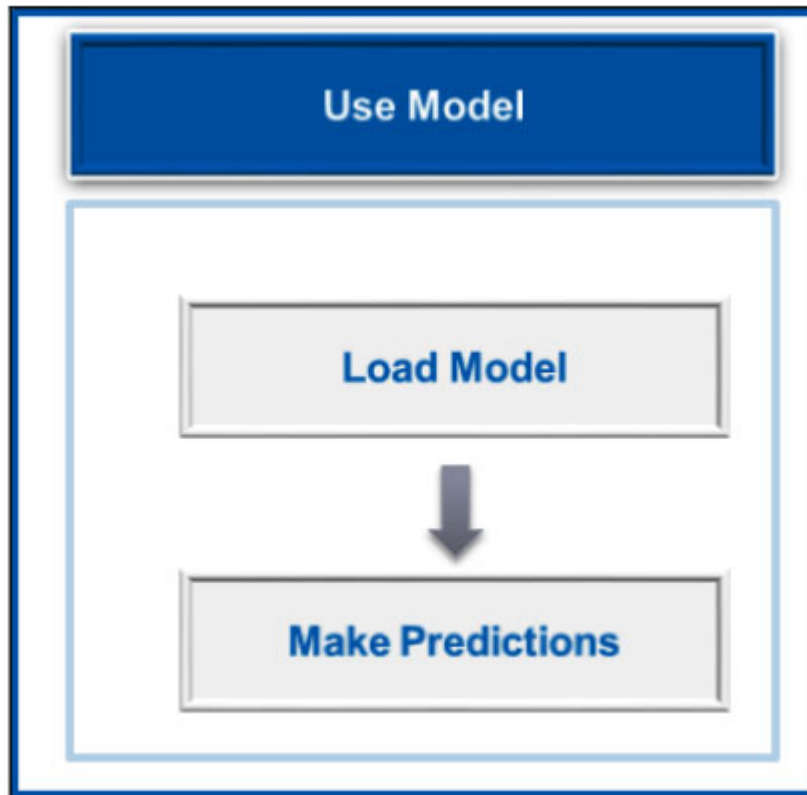
Once the training is complete, we can use the Save method to save the trained model to a file leveraging the DataViewSchema of the input data.

There are various ways to improve the model accuracy:

- **Reframe the problem:** Reframing the problem statement itself might improve the model accuracy.
- **Provide more data samples:** Providing more data samples improves the model performance.
- **Add context to the data:** Creating the appropriate context around the data points helps algorithms perform better.

- **Use meaningful data and features:** Add data and features that help reduce noise using techniques like **Permutation Feature Importance (PFI)**.

### Step 5: Using the Model



*Figure 7.8: Using the Model*

**Load Model:** We can load models that are stored locally or remotely. For locally stored models, we have to use the path of the file as input while for a model which is stored remotely, we have to use a Stream.

**Make Predictions:** The prediction pipeline contains both the trained model as well as the data pre-processing transformations. We can make single or batch predictions. We will now look at some of the samples provided by ML.NET.

## Samples

We have many samples based on ML.NET which are available which include:

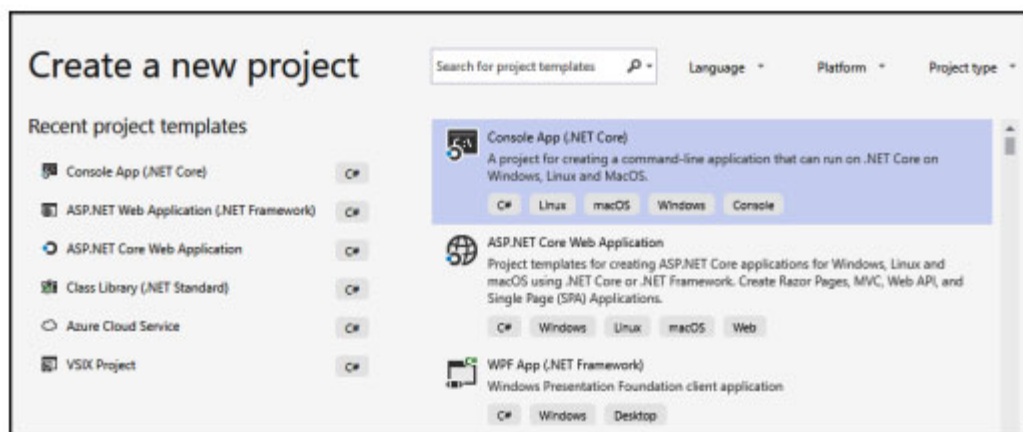
Issue Classification	Forecasting
Sentiment Analysis	Predictive maintenance
Image classification	Recommendations
Object detection	Customer segmentation

*Figure 7.9: Samples*

## Lab 1 - Creating Samples using ML.NET - Sentiment Analysis

In this lab we will create a sample application which will take input and perform sentiment analysis and return the result.

**Step 1:** Create a .NET Core Console Project. Open Visual Studio. Select **File | New | Project** from the menu bar. In the New Project dialog, select the Visual C# node followed by the .NET Core node. Then select the **Console App (.NET Core)** project template.



*Figure 7.10: Create New Project*

In the **Name** text box, type “**AzureAI\_SentimentAnalysis**” and then select the **OK** button.

**Configure your new project**

Console App (.NET Core) C# Linux macOS Windows Console

Project name

Location

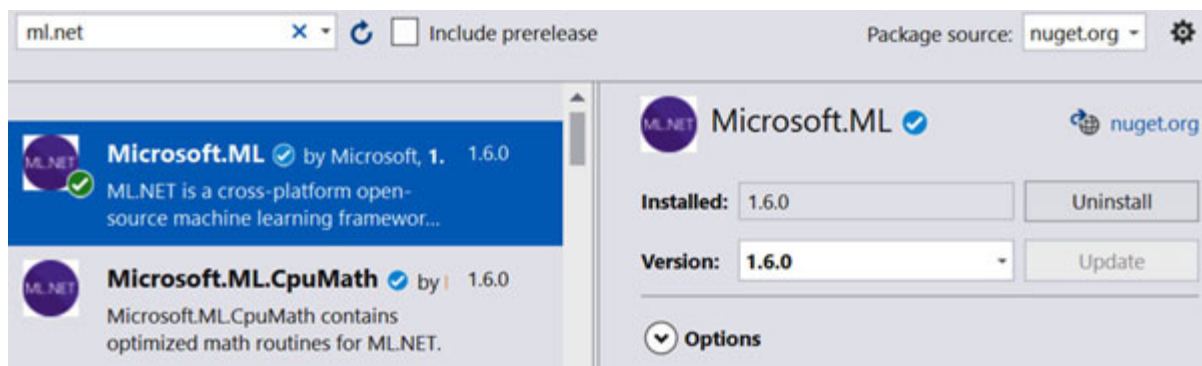
Solution

Solution name ⓘ

☒ Place solution and project in the same directory

*Figure 7.11: Configure Project*

**Step 2:** Install ML.NET Nuget Package by going to ‘Manage Nuget’ packages.



*Figure 7.12: Install ML.NET Nuget Package*

**Step 3:** Add a model class for sentiment. Sample code is given as follows:

```
internal class AzureAISentiment
{
    [LoadColumn(0)] public string InputSentimentText;
    [LoadColumn(1)] [ColumnName("Label")]
    public bool DetectedSentiment;
}
```



Add a new class **AzureAISentimentPrediction** for output which will be used as the prediction class used after model training. **PredictedValue** represents a Boolean value that represents the sentiment in the string that is passed as input. **PredictedScore** represents the raw score calculated by the model **PredictedProbability** – represents the calibrated score if it represents a positive score.

```
public class AzureAISentimentPrediction : AzureAISentiment
{
    [ColumnName("PredictedLabel")]
    public bool PredictedValue { get; set; }
    public float PredictedProbability
    {
        get; set;
    }
    public float PredictedScore
    {
        get; set;
    }
}
```

**Step 4:** Initializing MLContext: MLContext is the core object in ML.NET which is used to create a new ML.NET context which will be shared across the objects.

The following code snippet initializes the MLContext object.

```
MLContext mlContext = new MLContext();
```

**Step 5:** Data Preparation and loading

**Step 5.1:** Download sample data file.  
[https://github.com/mo20096258/AzureAI\\_SentimentAnalysis/blob/main/yelp\\_labelled.txt](https://github.com/mo20096258/AzureAI_SentimentAnalysis/blob/main/yelp_labelled.txt)

**Step 5.2:** Set the data Path

```
static readonly string testfilePath =
    Path.Combine(Environment.CurrentDirectory, "Data",
        "yelp_labelled.txt");
```

**Step 5.3:** We will use **IDataView** class for loading data from an input text file which contains two columns. First column represents the input

sentiment while the second column represents the sentiment. Create the **LoadSentimentData** method for loading the test data and splitting the dataset into train and test datasets.

- It loads the sentiment data.
- It splits the dataset into train and test datasets.
- It returns the split train and test datasets.

```
private static DataOperationsCatalog.TrainTestData
LoadSentimentData(IDataView
dataView, MLContext mlContext)
{ //train test data TrainTestData
DataOperationsCatalog.TrainTestData splitDataView =
mlContext.Data.TrainTestSplit(dataView, 0.3);
return splitDataView;
}
```

**Step 6:** Build And train the model the **BuildAndTrainModel()** method executes the following tasks:

- Extracts and transforms the data.
- Trains the model.
- Predicts sentiment based on test data.
- Returns the model.

```
private static ITransformer BuildAndTrainModel(MLContext
mlContext, IDataView trainingData)
{
var dataProcessPipeline =
mlContext.Transforms.Text.FeaturizeText("Features",
nameof(AzureAISentiment.Input.SentimentText));
//Set the training algorithm, then create and config the
modelBuilder
var trainer =
mlContext.BinaryClassification.Trainers.SdcaLogisticRegres
sion("Label", "Features");
var trainingPipeline =
dataProcessPipeline.Append(trainer);
```

```

//Train the model fitting to the DataSet ITransformer
trainedModel = trainingPipeline.Fit(trainingData);
return trainedModel;
}

```

**Step 7:** Now that the Model is built we have to provide the sentiment text for prediction.

```

private static AzureAISentimentPrediction predict(MLContext
mlContext, ITransformer trainedModel, AzureAISentiment
aiSentiment)
{
// Create prediction engine related to the loaded trained
model
var predEngine =
mlContext.Model.CreatePredictionEngine<AzureAISentiment,
AzureAISentimentPrediction>(trainedModel); // Score
var resultprediction = predEngine.Predict(aiSentiment);
return resultprediction;
}

```

**Code snippet for Main method call :**

```

public static void Main(string[] args)
{
int ch = 1;
do
{
MLContext mlContext = new MLContext();
IDataView dataView =
mlContext.Data.LoadFromTextFile<AzureAISentiment>
(testfilePath, hasHeader: false);
var trainTestSplit = LoadSentimentData(dataView, mlContext);
IDataView trainingData = trainTestSplit.TrainSet; IDataView
testData = trainTestSplit.TestSet; ITransformer trainedModel =
BuildAndTrainModel(mlContext, trainingData);
Console.WriteLine("\nPlease enter the sentiment text");
AzureAISentiment aiSentiment = new AzureAISentiment {
InputSentimentText = Console.ReadLine() };
}
}

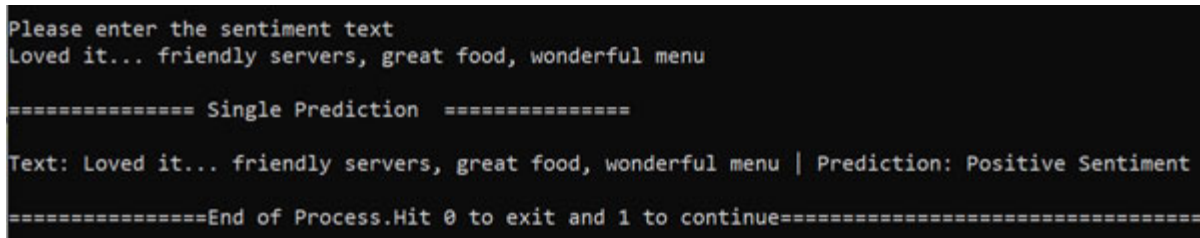
```

```

var resultprediction = predict(mlContext, trainedModel,
aiSentiment);
Console.WriteLine($"\\n===== Single Prediction
=====\\n"); Console.WriteLine( $"Text:
{aiSentiment.InputSentimentText} | Prediction:
{(Convert.ToBoolean(resultprediction.PredictedValue) ?
"Positive" : "Negative")} Sentiment ");
Console.WriteLine($"\\n=====End of Process.Hit 0 to
exit and 1 to continue=====\\n");
ch = Convert.ToInt32(Console.ReadLine());
}
while (ch == 1);
}

```

Sample output is as follows:



```

Please enter the sentiment text
Loved it... friendly servers, great food, wonderful menu

===== Single Prediction =====

Text: Loved it... friendly servers, great food, wonderful menu | Prediction: Positive Sentiment

=====End of Process.Hit 0 to exit and 1 to continue=====

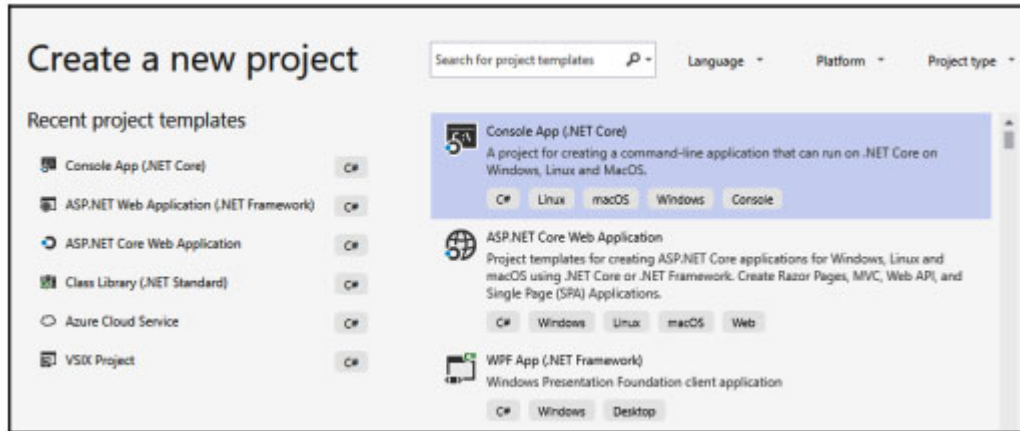
```

*Figure 7.13: Sample output*

## **Lab 2 - Creating samples using ML.NET - Fare prediction**

In this lab, we will create a sample application for predicting fare. We will create a console application which will build and train the model and predict the fare for the given input parameters.

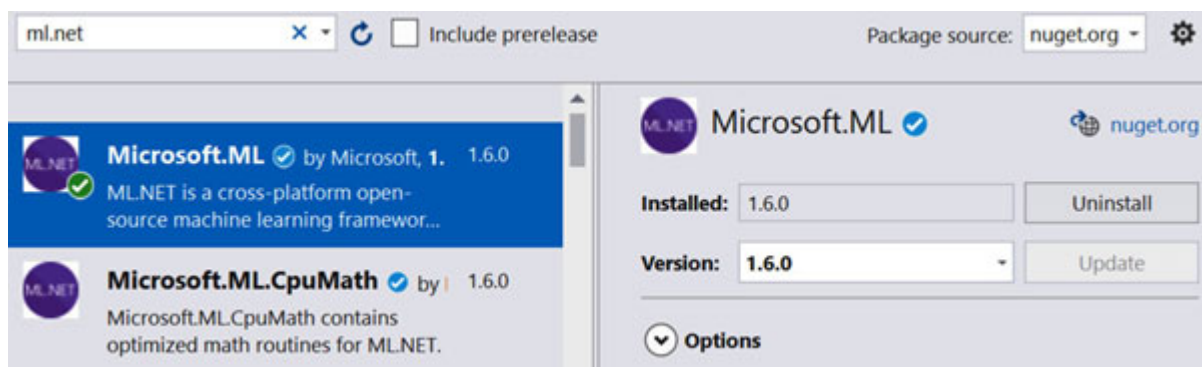
**Step 1:** Create a .NET Core Console Project. Open Visual Studio. Select **File | New | Project** from the menu bar. In the New Project dialog, select the Visual C# node followed by the .NET Core node. Then select the Console App (.NET Core) project template.



*Figure 7.14: Create Project*

In the **Name** text box, type “**AzureAIFarePrediction**” and then select the **OK** button.

Install ML.NET Nuget Package by going to ‘Manage Nuget packages.’



*Figure 7.15: Manage Nuget Packages Screen*

**Step 2:** Let us create classes for input data and predictions.

Add two new classes to your project: **AzureAIFarePrediction.cs** and **AzureAIFareData.cs**

```
{
    [ColumnName("Score")]
    public float FareAmount;
}

public class AzureAIFareData
{
    [LoadColumn(0)] public string VendorId; [LoadColumn(1)]
    public string RateCode;
```

```

[LoadColumn(2)] public float PassengerCount;
[LoadColumn(3)] public float TripTime;
[LoadColumn(4)] public float TripDistance; [LoadColumn(5)]
public string PaymentType;
[LoadColumn(6)] public float FareAmount;
}
public class AzureAIFarePrediction

```

**Step 3: Initializing MLContext:** MLContext is the core object in ML.NET which is used to create a new ML.NET context which will be shared across the objects. The following code snippet initializes the MLContext object:

```
MLContext mlContext = new MLContext();
```

**Step 4: Data Preparation and loading**

We can download the sample from below [https://github.com/mo20096258/AzureAI\\_SentimentAnalysis/blob/main/taxi-fare.csv](https://github.com/mo20096258/AzureAI_SentimentAnalysis/blob/main/taxi-fare.csv)

**Step 5: Build and train the Model**

The **BuildAndTrainModel()** method executes the following tasks: Extracts and transforms the data.

```

private static ITransformer BuildAndTrainModel(MLContext
mlContext, IDataView trainingData)
{
var dataProcessPipeline =
mlContext.Transforms.CopyColumns(outputColumnName: "Label",
inputColumnName: nameof(AzureAIFareData.FareAmount))
.Append(mlContext.Transforms.Categorical.OneHotEncoding(output
ColumnName: "VendorIdEncoded", inputColumnName:
nameof(AzureAIFareData.VendorId)))
.Append(mlContext.Transforms.Categorical.OneHotEncoding(output
ColumnName: "RateCodeEncoded", inputColumnName:
nameof(AzureAIFareData.RateCode)))
.Append(mlContext.Transforms.Categorical.OneHotEncoding(output
ColumnName: "PaymentTypeEncoded", inputColumnName:
nameof(AzureAIFareData.PaymentType)))
.Append(mlContext.Transforms.NormalizeMeanVariance(outputColum
nName: nameof(AzureAIFareData.PassengerCount)))
.Append(mlContext.Transforms.NormalizeMeanVariance(outputColum

```

```

nName: nameof(AzureAIFareData.TripTime)))
.Append(mlContext.Transforms.NormalizeMeanVariance(outputColumn
nName: nameof(AzureAIFareData.TripDistance)))
.Append(mlContext.Transforms.Concatenate("Features",
"VendorIdEncoded", "RateCodeEncoded", "PaymentTypeEncoded",
nameof(AzureAIFareData.PassengerCount) ,
nameof(AzureAIFareData.TripTime),
nameof(AzureAIFareData.TripDistance)));
Set the training algorithm, then create and config the
modelBuilder var trainer =
mlContext.Regression.Trainers.Sdca(labelColumnName: "Label",
featureColumnName: "Features"); var trainingPipeline =
dataProcessPipeline.Append(trainer); // STEP 4: Train the
model fitting to the DataSet ITransformer trainedModel =
trainingPipeline.Fit(trainingData); return trainedModel;
}

```

## Step 6: Predict for sample data

```

private static AzureAIFarePrediction predict(MLContext
mlContext, ITransformer trainedModel, AzureAIFareData
aiFareData)
{
var predEngine =
mlContext.Model.CreatePredictionEngine<AzureAIFareData,
AzureAIFarePrediction>(trainedModel);
// Prediction
var resultprediction = predEngine.Predict(aiFareData);
return resultprediction;
}

```

## Step 7: Print output Console.WriteLine(\$"\\n===== Single Prediction =====\\n");

```

Console.WriteLine($"VendorId: {taxiTripSample.VendorId} |
RateCode:{taxiTripSample.RateCode}\\n" + $"PassengerCount:
{taxiTripSample.PassengerCount} | TripTime:
{taxiTripSample.TripTime}\\n" + $"TripDistance:
{taxiTripSample.TripDistance} | PaymentType:
{taxiTripSample.PaymentType}\\n" + $"FareAmount:
{taxiTripSample.FareAmount}\\n"); Console.WriteLine($"Predicted

```

```
fare: {resultprediction.FareAmount:0.####}, actual fare:
15.5");
```

Sample Code snippet of main method

```
public static void Main(string [] args)
{ int ch = 1;
do {
MLContext mlContext = new MLContext();
IDataView dataView =
mlContext.Data.LoadFromTextFile<AzureAIFareData>
(datasetfilePath, hasHeader: true, separatorChar: ',');
var count = dataView.GetColumn<float>
(nameof(AzureAIFareData.FareAmount)).Count(); IDataView
trainingDataView = mlContext.Data.FilterRowsByColumn(dataView,
nameof(AzureAIFareData.FareAmount), lowerBound: 1, upperBound:
150); var count2 = trainingDataView.GetColumn<float>
(nameof(AzureAIFareData.FareAmount)).Count(); var
trainTestSplit = mlContext.Data.TrainTestSplit(dataView, 0.2);
IDataView trainingData = trainTestSplit.TrainSet; IDataView
testData = trainTestSplit.TestSet; ITransformer trainedModel =
BuildAndTrainModel(mlContext, trainingData);
//Console.WriteLine("\nPlease enter the sentiment text"); var
taxiTripSample = new AzureAIFareData() { VendorId = "VTS",
RateCode = "1", PassengerCount = 1, TripTime = 1140,
TripDistance = 3.75f, PaymentType = "CRD", FareAmount = 0 //
To predict. Actual/Observed = 15.5 }; var resultprediction =
predict(mlContext, trainedModel, taxiTripSample);
Console.WriteLine($"\\n===== Single Prediction
=====\\n"); Console.WriteLine($"VendorId:
{taxiTripSample.VendorId} | RateCode:
{taxiTripSample.RateCode}\\n" + $"PassengerCount:
{taxiTripSample.PassengerCount} | TripTime:
{taxiTripSample.TripTime}\\n" + $"TripDistance:
{taxiTripSample.TripDistance} | PaymentType:
{taxiTripSample.PaymentType}\\n" + $"FareAmount:
{taxiTripSample.FareAmount}\\n"); Console.WriteLine($"Predicted
fare: {resultprediction.FareAmount:0.####}, actual fare:
15.5"); Console.WriteLine($"\\n=====End of
```



```

Process.Hit 0 to exit and 1 to
continue=====\\n");
    ch = Convert.ToInt32(Console.ReadLine());
} while (ch == 1);
}

```

The following is a snapshot of the output:

```

===== Single Prediction =====

VendorId: VTS | RateCode:1
PassengerCount:1 | TripTime:1140
TripDistance:3.75 | PaymentType:CRD
FareAmount:0

Predicted fare: 15.6238, actual fare: 15.5

=====End of Process.Hit 0 to exit and 1 to continue=====

```

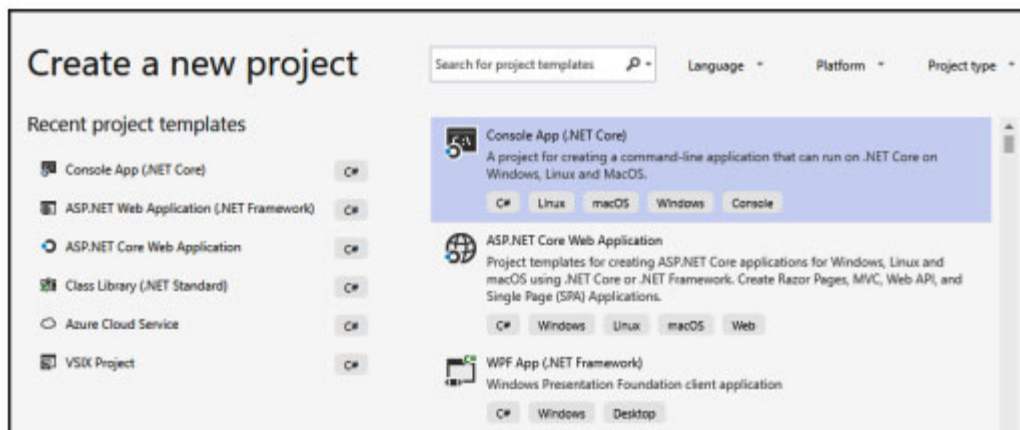
*Figure 7.16: Sample Output*

## Lab 3 - Creating samples using ML.NET - Issue classification

In this lab, we will create a sample console application which will leverage ML.NET model creation and training to classify issues.

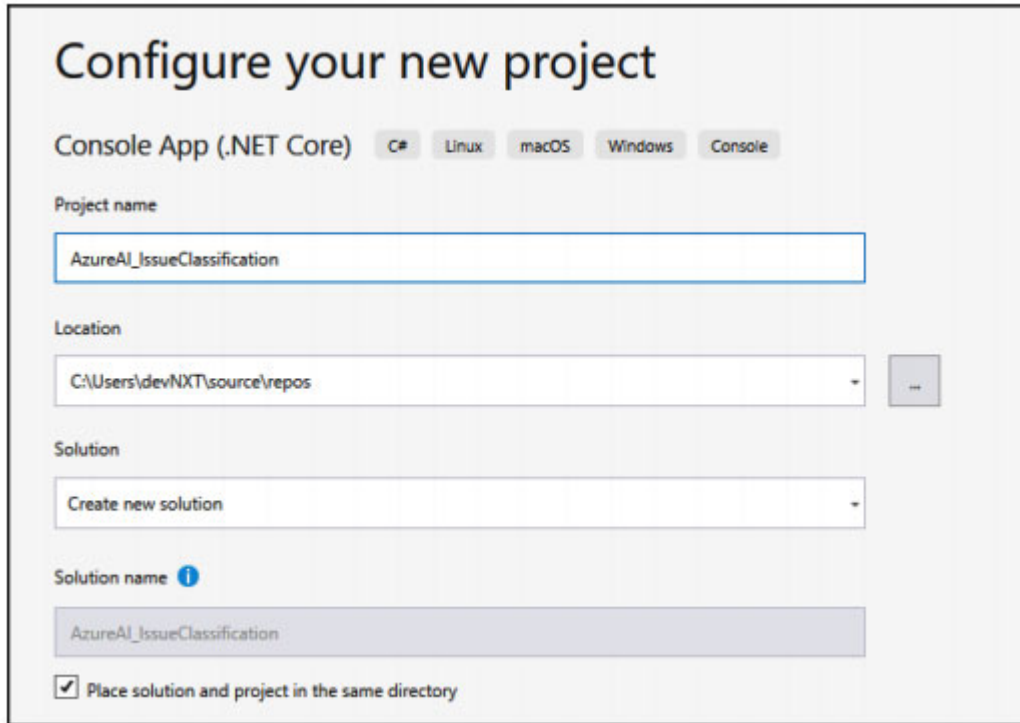
**Step 1:** Create a .NET Core console project.

Open Visual Studio. Select **File** | **New** | **Project** from the menu bar. In the New Project dialog, select the Visual C# node followed by the .NET Core node. Then select the **Console App (.NET Core)** project template



*Figure 7.17: Create Project*

In the **Name** text box, type “**AzureAI\_IssueClassification**” and then select the **OK** button.



Configure your new project

Console App (.NET Core) C# Linux macOS Windows Console

Project name  
AzureAI\_IssueClassification

Location  
C:\Users\devNXT\source\repos

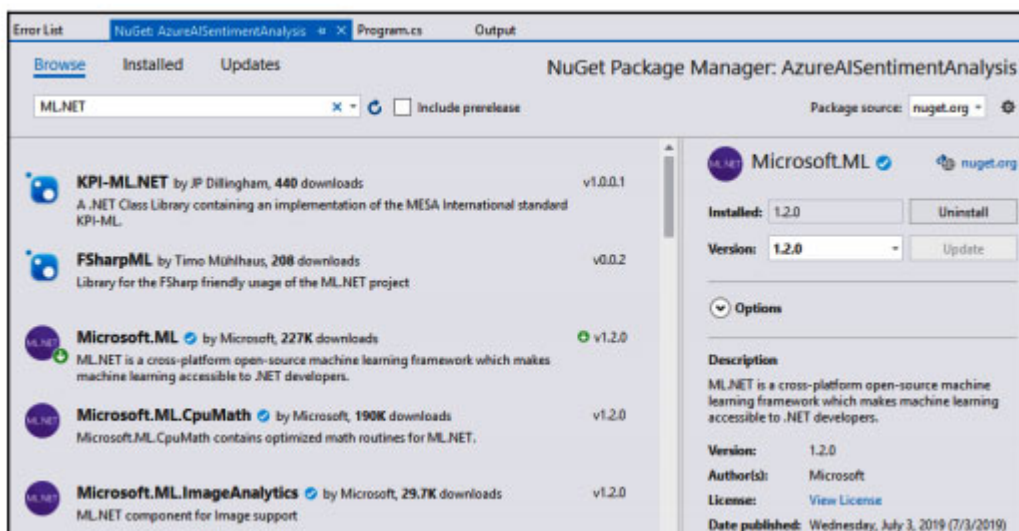
Solution  
Create new solution

Solution name ⓘ  
AzureAI\_IssueClassification

☒ Place solution and project in the same directory

*Figure 7.18: Configure Project*

Install ML.NET Nuget package by going to Manage Nuget packages.



*Figure 7.19: Install ML.NET Nuget Package*

**Step 2:** Let us create classes for input data and predictions.

Add two new classes to your project: **AzureAI\_IssuePrediction.cs** and **AzureAI\_Issue.cs**

```
public class AzureAI_IssuePrediction
{
    [ColumnName("PredictedLabel")] public string Area;
}

public class AzureAI_Issue
{
    [LoadColumn(0)] public string ID { get; set; }
    [LoadColumn(1)] public string Area { get; set; }
    [LoadColumn(2)] public string Title { get; set; }
    [LoadColumn(3)] public string Description { get; set; }
}
```

**Step 3:** Initializing MLContext: MLContext is the core object in ML.NET which is used to create a new ML.NET context which will be shared across the objects.

The following code snippet initializes the MLContext object. **MLContext mlContext = new MLContext();**

**Step 4:** Data Preparation and loading

We can download the sample from below [https://github.com/mo20096258/AzureAI\\_SentimentAnalysis/blob/main/corefx\\_issues.tsv](https://github.com/mo20096258/AzureAI_SentimentAnalysis/blob/main/corefx_issues.tsv)

**Step 5:** Build and train the model the **BuildAndTrainModel()** method executes the following tasks: Extracts and transforms the data.

```
private static ITransformer BuildAndTrainModel(MLContext
mlContext, IDataView trainingData)
{
    var dataProcessPipeline =
    mlContext.Transforms.Conversion.MapValueToKey(outputColumnName
: "Label", inputColumnName: nameof(AzureAI_Issue.Area))
    .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnNa
me: "TitleFeaturized", inputColumnName:
nameof(AzureAI_Issue.Title)))
    .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnNa
me: "DescriptionFeaturized", inputColumnName:
nameof(AzureAI_Issue.Description)))
}
```

```

.Append(mlContext.Transforms.Concatenate(outputColumnName:
"Features", "TitleFeaturized", "DescriptionFeaturized"))
.AppendCacheCheckpoint(mlContext); // STEP 3: Set the training
algorithm, then create and config the modelBuilder
IEstimator<ITransformer> trainer =
mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy
("Label", "Features"); var trainingPipeline =
dataProcessPipeline.Append(trainer)
.Append(mlContext.Transforms.Conversion.MapKeyToValue("Predict
edLabel")); // STEP 4: Train the model fitting to the DataSet
ITransformer trainedModel =
trainingPipeline.Fit(trainingData); return trainedModel;
}

```

### Predict the result:

```

private static AzureAI_IssuePrediction predict(MLContext
mlContext, ITransformer trainedModel, AzureAI_Issue aiIssue)
{ var predEngine =
mlContext.Model.CreatePredictionEngine<AzureAI_Issue,
AzureAI_IssuePrediction>(trainedModel); // Prediction var
resultprediction = predEngine.Predict(aiIssue); return
resultprediction;
}

```

### Step 6: Evaluate the model.

```

private static void EvaluateModel(MLContext mlContext,
IDataView testData, ITransformer trainedModel)
{ var testMetrics =
mlContext.MulticlassClassification.Evaluate(trainedModel.Trans
form(testData)); //Print the metrics Console.WriteLine($"*
Metrics for Multi-class Classification model - Test Data ");
Console.WriteLine($"*****
*****
*****");
Console.WriteLine($"* MicroAccuracy:
{testMetrics.MicroAccuracy:0.###}"); Console.WriteLine($"*
MacroAccuracy: {testMetrics.MacroAccuracy:0.###}");
Console.WriteLine($"* LogLoss: {testMetrics.LogLoss:#.###}");
Console.WriteLine($"* LogLossReduction:

```

```

{testMetrics.LogLossReduction:###}");
Console.WriteLine($"*****
*****
*****");
}

```

### Code snippet for **Main** method:

```

public static void Main(string[] args)
{ int ch = 1; do {
MLContext mlContext = new MLContext(); IDataView dataView =
mlContext.Data.LoadFromTextFile<AzureAI_Issue>
(datasetfilePath, hasHeader: true, separatorChar: '\t',
allowSparse: false); var trainTestSplit =
mlContext.Data.TrainTestSplit(dataView, 0.2); IDataView
trainingData = trainTestSplit.TrainSet; IDataView testData =
trainTestSplit.TestSet; ITransformer trainedModel =
BuildAndTrainModel(mlContext, trainingData);
EvaluateModel(mlContext, testData, trainedModel);
//Console.WriteLine("\nPlease enter the sentiment text"); var
aiIssue = new AzureAI_Issue {
ID = "25",
Title = "WebSockets communication is slow in my machine",
Description = "The WebSockets communication used under the
covers by SignalR looks like is going slow in my development
machine.." };
var resultprediction = predict(mlContext, trainedModel,
aiIssue); Console.WriteLine($"\\n===== Single
Prediction =====\\n"); Console.WriteLine($"Title:
{aiIssue.Title}\\nDescription: {aiIssue.Description} \\n\\n" +
$"Prediction-Result: {resultprediction.Area} ");
Console.WriteLine($"\\n=====End of Process.Hit 0 to
exit and 1 to continue=====\\n");
ch = Convert.ToInt32(Console.ReadLine()); } while (ch == 1); }

```

Following is the output:

```

* Metrics for Multi-class Classification model - Test Data
*****
* MicroAccuracy: 0.702
* MacroAccuracy: 0.467
* Logloss: 1.22
* LoglossReduction: .506
*****

***** Single Prediction *****

Title: WebSockets communication is slow in my machine
Description: The WebSockets communication used under the covers by SignalR looks like is going slow in my development machine..
Prediction-Result: area-System.Net

*****End of Process.Hit 0 to exit and 1 to continue*****

```

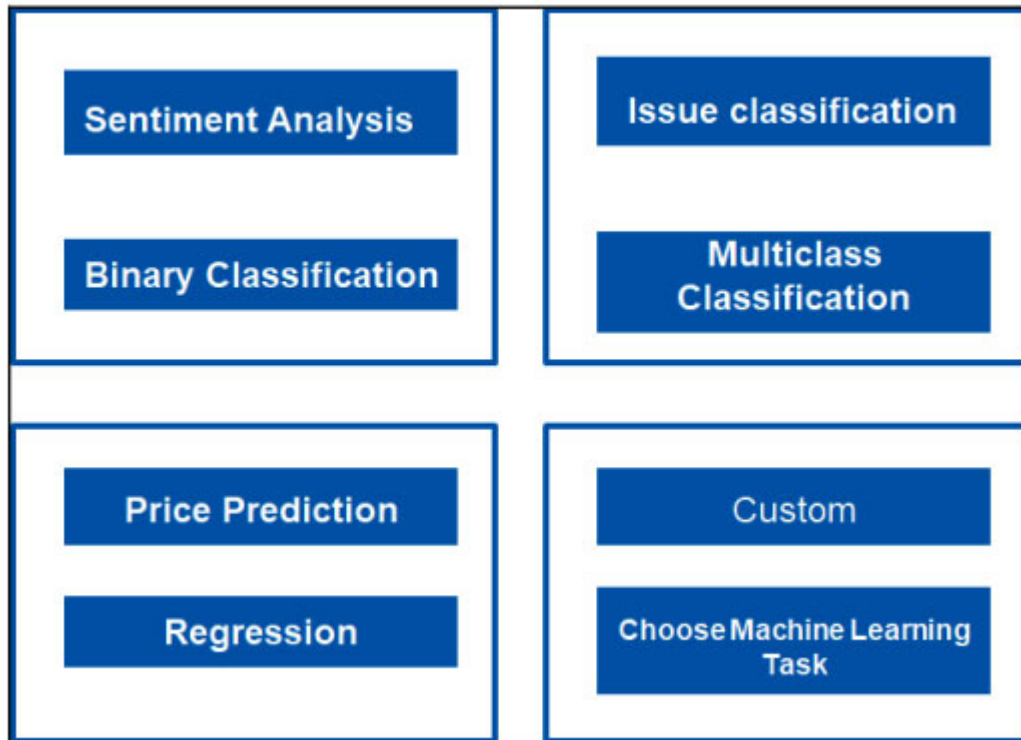
*Figure 7.20: Sample output*

## **ML.NET Model Builder**

Model Builder helps developers build models with a graphical user friendly interface without much Machine Learning expertise. Model Builder supports a number of scenarios like classifying sentiments, detecting fraudulent transactions, classifying issues, and so on.

### **Templates**

Model Builder comes with templates which can be used as starting points for common scenarios like sentiment analysis, issue classification, price prediction, and custom scenarios.



*Figure 7.21: Model Builder Templates*

## Data

Once we choose the template, we need to input the dataset that will be used by the model and the output that needs to be predicted.

A dataset has features or attributes as columns and sample data as rows. For instance, a bicycle price estimator will have features like model, make, year of manufacturing etc. as features and price as label. Model Builder places the following limitations on the data:

- Data can be read from a (.csv or .tsv with a header row)
- Data can be read from SQL server database-limit of 100000 rows
- Data copied from SQL server to local machine before training

## Train

Once the dataset is loaded, it needs to be trained. The exact time to train depends on:

- Columns type's - whether numeric or text Task type –

- Regression and so on
- Rows for training
- Number of feature columns

Evaluate evaluation measures how good the predictions using new data against the model that has been trained. Model Builder splits the training data into a training (80%) set and a test set (20%).

Improve if your model performance score is not as good as expected, we need to train for a longer duration, or add more data or balance the data. For instance, if some tags are not used effectively, unbalanced data will make the model struggle to predict.

**Model File:** After the evaluation phase, code that can be added as a model file is output which can be added to an application.

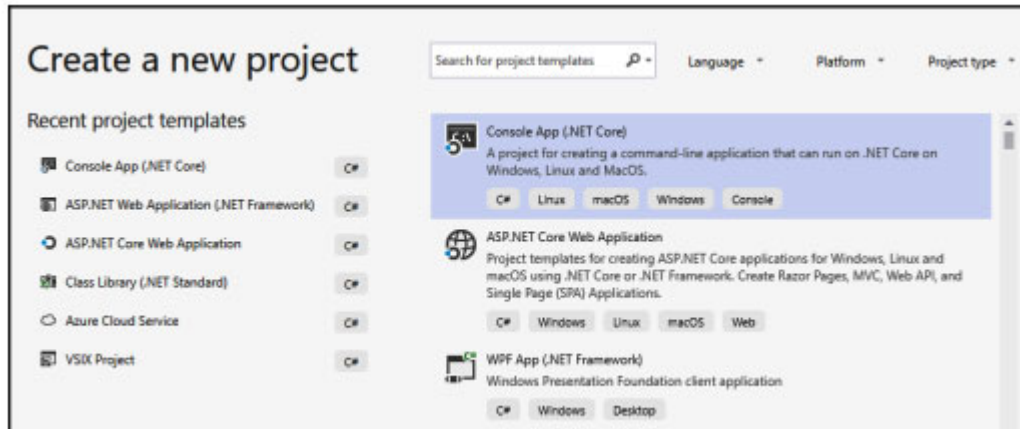
## **Lab 4 - Creating sample using ML.NET Model Builder for sentiment analysis**

In this lab we will create a sample application which will take input and perform sentiment analysis using ML.NET Model Builder and return the result.

**Step 1:** Go to <http://aka.ms/mlnettemplates> for Visual Studio and download ML.NET Model Builder extension.

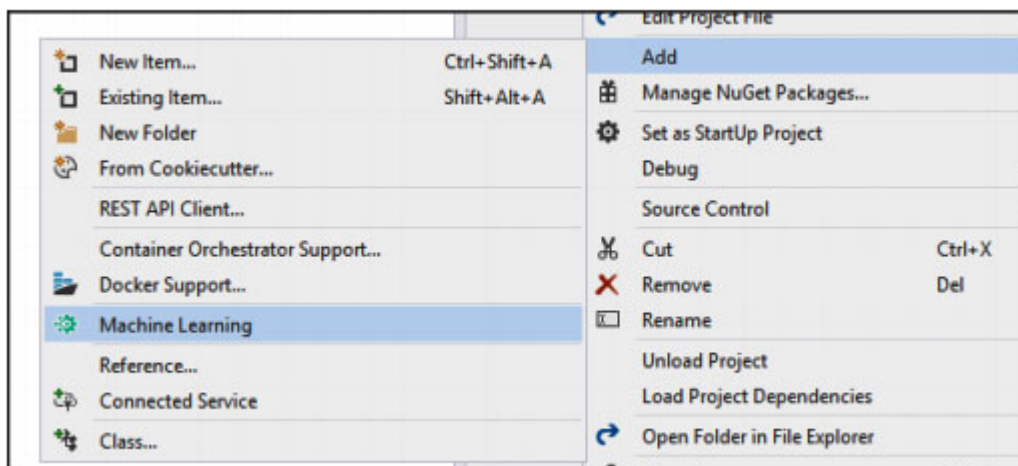
**Step 2:** Create a .NET Core Console Project. Open Visual Studio. Select **File | New | Project** from the menu bar. In the New Project dialog, select the Visual C# node followed by the .NET Core node. Then select the Console App (.NET Core) project template.





*Figure 7.22: Create Project*






**Step 3:** Right click **Solution Explorer**. Select **Add | Machine Learning**.



*Figure 7.23: Add Package*

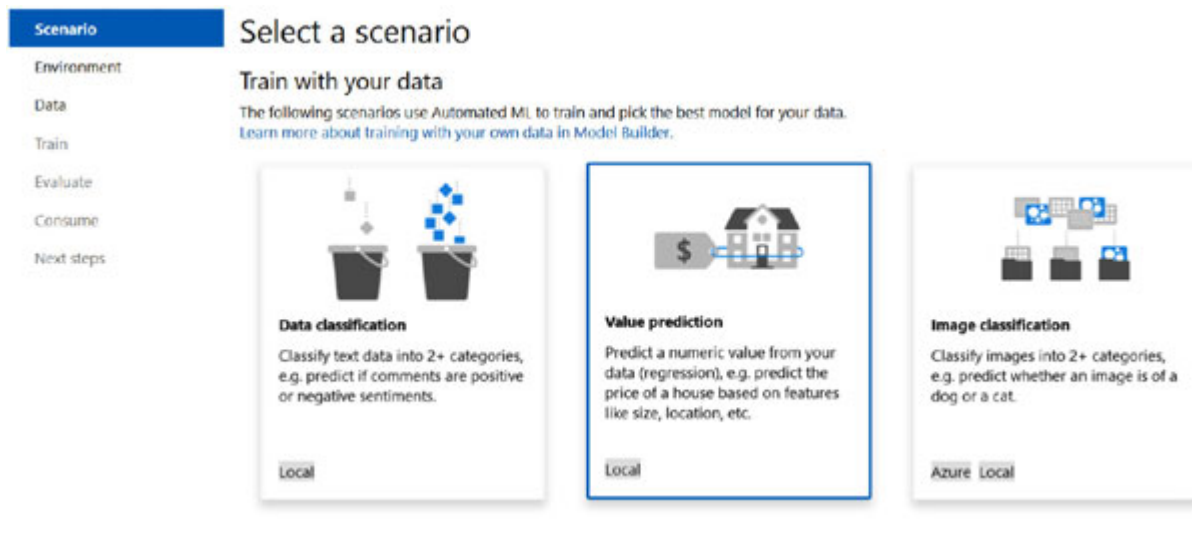
**Step 4:** Choose a Data Model for this sample we have downloaded yelp from UCI Sentiment Labeled Sentences dataset. We can download from below link

<https://archive.ics.uci.edu/ml/machine-learning-databases/00331/sentiment%20labelled%20sentences.zip>).

Name	Type	Compressed size
 .DS_Store	DS_STORE File	1 KB
 amazon_cells_labelled	Text Document	23 KB
 imdb_labelled	Text Document	34 KB
 readme	Text Document	1 KB
 yelp_labelled	Text Document	24 KB

*Figure 7.24: Choose Data Model*

Right click on `yelp_labelled.txt` and save as `yelp_labelled.tsv`.  
Choose the Model Type:



*Figure 7.25: Choose Model Type*

Choose your Dataset for training the model and select the column to predict.

Scenario
Environment
**Data**
Train
Evaluate
Consume
Next steps

## Add data

In order to build a model, you must add data and choose your column to predict.  
[How do I get sample datasets and learn more?](#)

### Input

Data source type

☒ File (.csv, .tsv, .txt)
☐ SQL Server

C:\Users\sentiment labelled sentences\sentiment labelled sentences\yelp\_labelled.tsv
Browse...

Column to predict (Label): ⓘ

col1

[Advanced data options...](#)

### Data Preview

10 of 1,000 rows, and all 2 columns (including 0 columns that are ignored).

col0	col1
Wow... Loved this place.	1
Crust is not good.	0
Not tasty and the texture was just nasty.	0

Feedback

**Figure 7.26: Add Data**

Train choose the right duration to train. We need to set larger durations for larger datasets. The training might fail when the durations are small.

Scenario
Environment
Data
**Train**
Evaluate
Consume
Next steps

## Train

Specify a time to train for evaluating various models.  
[How long should I train for?](#)

Training setup summary ▾

Time to train (seconds): ⓘ

Train again
✓ Training complete

### Training results

Best quality (RSquared): 0.24

Best model: FastForestRegression

Training time: 8.28 seconds

Models explored (total): 3

Generated code-behind: MLModel1.consumption.cs, MLModel1.training.cs

Next step

**Figure 7.27: Train Model**

Progress of the model training is displayed as follows:

- **Status:** shows model training process status.
- **Accuracy:** Accuracy of the best model.
- **Best algorithm:** Shows the best algorithm.
- **Last algorithm:** Shows the last algorithm that was explored by Model Builder.

### Step 5: Evaluate the model

Scenario

Environment

Data

Train

**Evaluate**

Consume

Next steps

## Evaluate

Results of training for your model can be found below.  
[How do I understand my model performance?](#)

**Best model:**

**RSquared:** 0.24

**Model:** FastForestRegression

### Try your model

**Sample data**

The following fields below are pre-filled by row 1 of your data.

col0

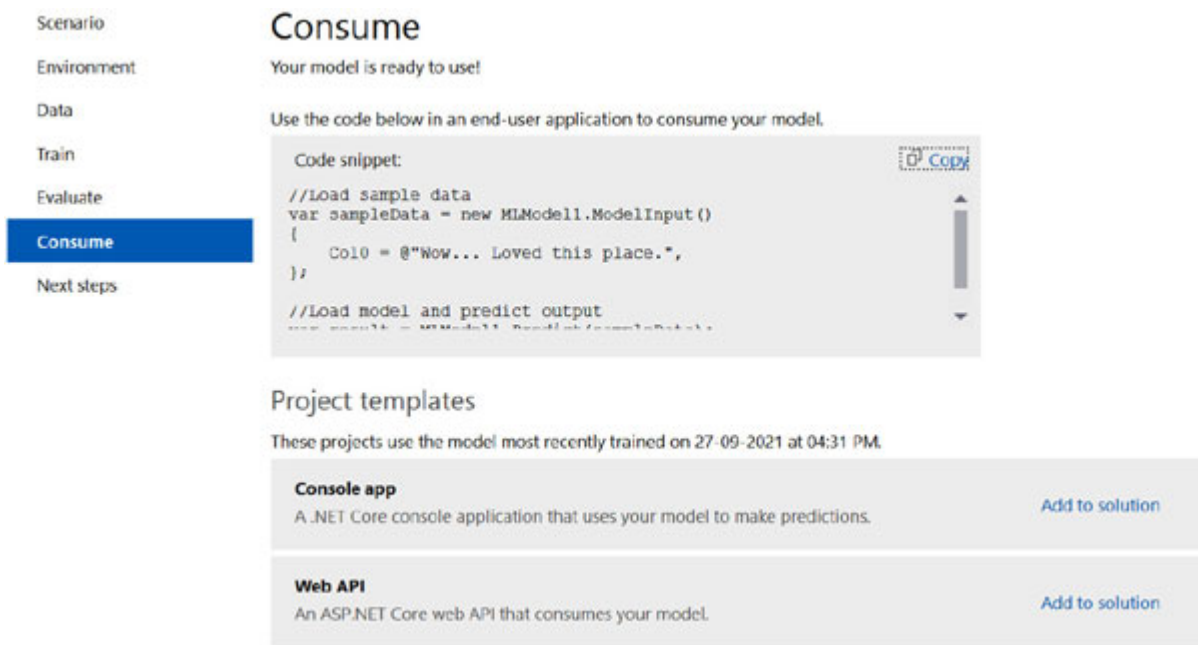
Wow... Loved this place.

Predict

Results	
col1:	0.84

*Figure 7.28: Evaluate Model*

**Step 6:** Trained model is ready for consumption.



*Figure 7.29: Model Ready for Consumption*

## Conclusion

In this chapter, we have looked at ML.NET basics, how to leverage ML.NET to develop models and infuse custom Machine Learning into applications, develop a sample ML Model leveraging Vision API, expose the ML model as a service and invoke the service from .NET Core client.

In the next chapter, we will look at Accord.NET.

## Multiple choice questions

This section consists of a set of questions that are designed to test your knowledge of the concepts covered in this chapter.

When answering these questions, imagine there is an audience who has watched a movie.

You have images of the movie and images of the audience coming out of the theatre.

### **1. We can convert raw data into features using**

- a. Analysers
- b. ML Extraction

- c. Data Transforms
  - d. Feature Engineering
- 2. Which algorithms create a model that calculate scores from a linear combination of the input data and a set of corresponding weights?**
- a. Linear Regression
  - b. KNN
  - c. Multiclass
  - d. Binary
- 3. Which algorithms are to discover grouping in the input data provided like grouping customers by segments?**
- a. Clustering
  - b. Regression
  - c. Multiclass
  - d. Binary
- 4. Which type of learning is applicable where some portion of the input data is labeled but most portion of it is unlabeled?**
- a. Supervised Learning
  - b. Unsupervised Learning
  - c. Semi Supervised Learning
  - d. All of the Above
- 5. Which type of learning is applicable to predict an output variable continuous value like predicting taxi fare for a given trip based on certain conditions**
- a. Clustering
  - b. Regression
  - c. Multiclass
  - d. Binary
- 6. Which trainer create a model that calculate scores from a linear combination of the input data and a set of corresponding weights.**

- a. Linear
  - b. Principal Component Analysis
  - c. K Means
  - d. Naïve Bayes
- 7. What framework capable of handling large data and is designed to be distributed and efficient with faster training speed, lower memory usage and higher accuracy and efficiency**
- a. Linear
  - b. Light Gradient Boosted Machine
  - c. Fast Tree
  - d. Decision Tree
- 8. Which technique is used to split the data into several partitions and trains multiple algorithms on these partitions to improve the robustness of the training process?**
- a. Validation
  - b. Principal Component Analysis
  - c. Cross Validation
  - d. Partitioning
- 9. What is the learning of the model with labelled input data and an algorithm to map the input data to the output data.**
- a. Supervised Learning
  - b. Unsupervised Learning
  - c. Semi Supervised Learning
  - d. All of the Above
- 10. What can we use to standardize features which are not on the same scale?**
- a. Validation
  - b. Normalization
  - c. Cross Validation
  - d. Partitioning

## **Answers**

1. **d. Feature Engineering**
2. **a. Linear Regression**
3. **a. Clustering**
4. **c. Semi Supervised Learning**
5. **b. Regression**
6. **a. Linear Trainers**
7. **b. Light Gradient Boosted Machine**
8. **b. Principal Component Analysis**
9. **a. Supervised Learning**
10. **b. Normalization**



# **CHAPTER 8**

## **Using Azure ML Studio**

In the last few chapters, we have read about various cognitive services available in Azure. We have also read about ML.NET and Applied Azure AI Services. This chapter focuses on how we can use Azure Machine Learning Studio using Automated ML and Designer. We will understand ML Ops and how to migrate from Classic Azure ML Studio. Towards the end of the chapter we will learn about Accord.Net Machine Learning Framework.

### **Structure**

The following topics will be covered in this chapter:

- Understanding Azure Machine Learning Studio
- Sample Lab using ML Studio Automated ML
- Sample Lab using ML Studio Azure Designer
- ML Ops - A brief overview
- Migration from Classic Azure ML Studio - A brief overview
- Azure ML library of algorithms
- Understanding Accord.Net Machine Learning Framework

### **Objective**

After reading this chapter, the reader will be able to leverage Azure Machine Learning Studio to create models, score models, and run experiments and also learn about Accord Framework.

### **Azure Machine Learning Studio**

Azure Machine Learning studio is a project authoring and asset management portal which provides no-code and code-first experiences for

creating and managing assets and resources like models, run logs, datastores, notebooks, experiments, pipelines, datasets and compute resources. From data ingestion to using the model as a service, to container orchestration, Azure Machine Learning Services helps in making it easier and faster.

The Azure Machine Learning studio provides different ways to author machine learning projects.

- **Notebooks:** We can leverage Jupyter notebooks. This has integration with Jupyter Notebook Servers.
- **Azure Machine Learning designer:** Azure Machine Learning Designer provides a no code based approach which helps in developing ML applications using simple drag and drop without writing any code.
- **Automated machine learning UI:** Supervised machine learning models like classification, regression and time series forecasting where training data and known labels are available are supported by the Automated machine learning studio. It finds the best model based on chosen metrics by iterating over combinations of and hyperparameters and algorithms.

Now we will get familiar with Azure Machine Learning Studio by working on two machine learning samples. We will go through sample labs using Azure Machine Learning Studio with two options one with Automated ML and one with Designer.

## [Sample lab using Azure ML Studio using Automated machine learning UI no-code approach](#)

Automated machine learning helps developers build ML models with large scale, efficiency, and productivity by automating time-consuming, iterative tasks of machine learning model development. Automated machine learning rapidly iterates over many combinations of algorithms and hyperparameters to help find the optimum model based on a success metric chosen.

**Problem Statement:** In this sample, we will use a dataset which is populated with data from a local dataset. We will create a workspace, load

the sample dataset, create an experiment and run the experiment. We will observe how the model returns results. In this lab, we will explore the No Code Option. We will use Azure Machine Learning automated ML to train a classification model, deploy and evaluate the model.

In this lab, we will perform the following steps:

**Step 1:** Create an Azure Machine Learning workspace.

**Step 2:** Load Dataset.

**Step 3:** Run an automated machine learning experiment using the Dataset.

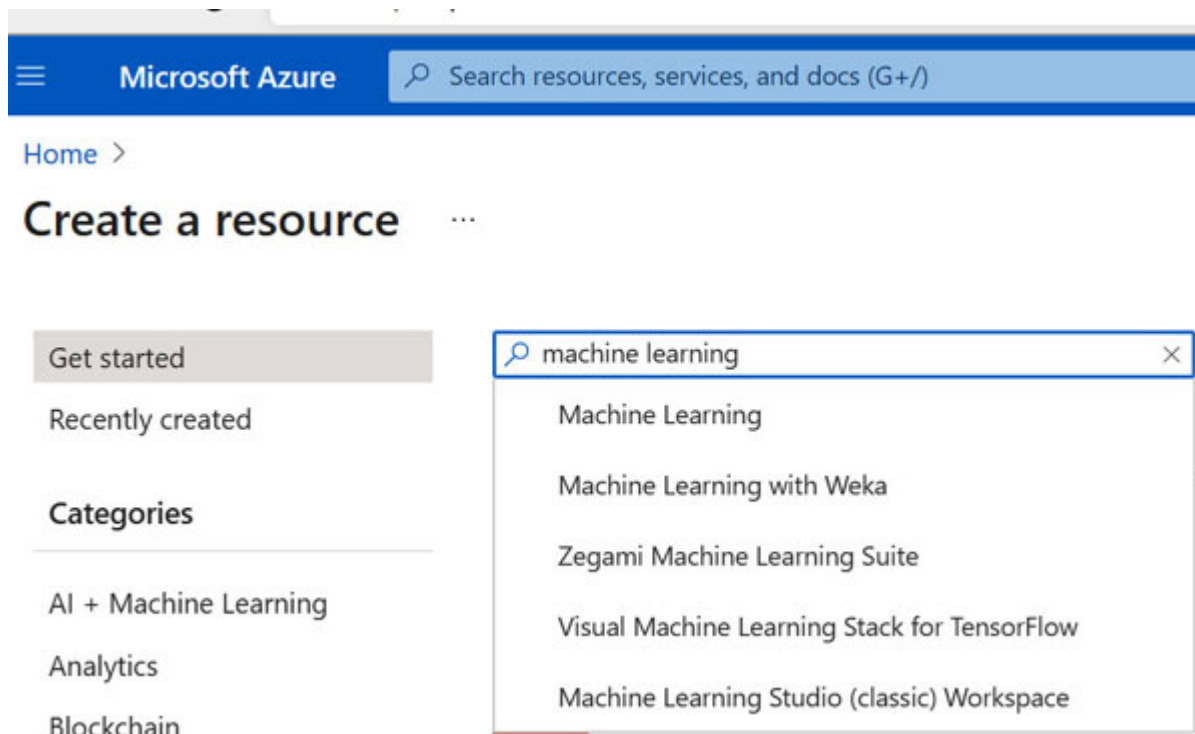
The following are the steps in detail:

**Step 1:** Create an Azure Machine Learning workspace.

A workspace is the foundational block for Azure machine learning project. We will follow the below steps to create a workspace.

Go to *portal.azure.com*.

Click on **Create Resource-Type Machine Learning**



**Figure 8.1:** Create Resource

Click on **create** and fill in the details to create a workspace.

# Machine learning ...

Create a machine learning workspace

✓ Validation passed

Basics   Networking   Advanced   Tags   Review + create

## Basics

Subscription	Free Trial
Resource group	(New) test14
Region	East US 2
Workspace name	MLStudioWS
Storage account	(new) mlstudiows7387060628
Key vault	(new) mlstudiows6120440310
Application insights	(new) mlstudiows3918018209
Container registry	(new) 123453242342testcontainer

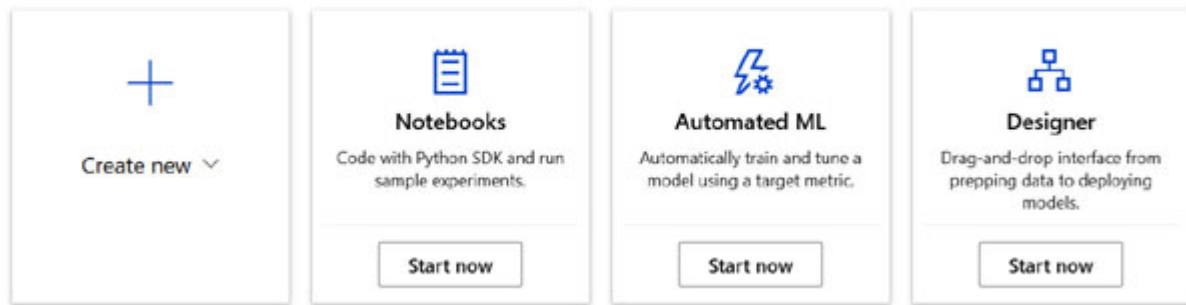
*Figure 8.2: Create Workspace*

After creating the workplace, let is go to Azure Machine Learning Studio to work on the next steps to create an experiment that leverages Automated ML.

**Step 2:** Navigate Azure Machine Learning Studio:

Go to <https://ml.azure.com/>

## Welcome to the Azure Machine Learning Studio



*Figure 8.3: Azure ML Studio*

Click on Automated ML: This will open up the following screen:

**Step 2:** Create and load dataset: We will need a dataset to create the experiment. The dataset can be chosen from existing local file or sample datasets available or we can create our own dataset.

Click on **Create dataset**. This will open the following screen:

The image shows the 'Create dataset from local files' screen in Azure ML Studio. The page has a light blue header with the title 'Create dataset from local files'. On the left side, there is a vertical navigation pane with five radio buttons: 'Basic info' (selected), 'Datastore and file selection', 'Settings and preview', 'Schema', and 'Confirm details'. The main area on the right is titled 'Basic info' and contains three sections: 'Name' with a text input field containing 'BankMarketingDataset', 'Dataset type' with a dropdown menu set to 'Tabular', and 'Description' with a text area containing 'Example Dataset'. At the bottom of the main area, there are two buttons: 'Back' and 'Next'.

*Figure 8.4: Create Dataset*

Choose 'Create from Local file'. Sample file can be downloaded from [here](#).

### Create dataset from local files

- Basic info
- Datastore and file selection**
- Settings and preview
- Schema
- Confirm details

#### Datastore and file selection

workspaceartifactstore

Create new datastore

Select files for your dataset \*

These files will be uploaded to your selected datastore and made available in your workspace. Supported file types include: delimited (i.e. csv, tsv), Parquet, JSON Lines, and plain text.

Upload

File name	Size (MiB)	Upload %	Status
bankmarketing_train.csv	3.958		

*Figure 8.5: Create Dataset*

Upload the file by choosing the file from the local store:

- Basic info
- Datastore and file selection**
- Settings and preview
- Schema
- Confirm details

#### Datastore and file selection

Select files for your dataset \*

These files will be uploaded to your selected datastore and made available in your workspace. Supported file types include: delimited (i.e. csv, tsv), Parquet, JSON Lines, and plain text.

Upload

File name	Size (MiB)	Upload %	Status
bankmarketing_train.csv	3.958		

Upload path

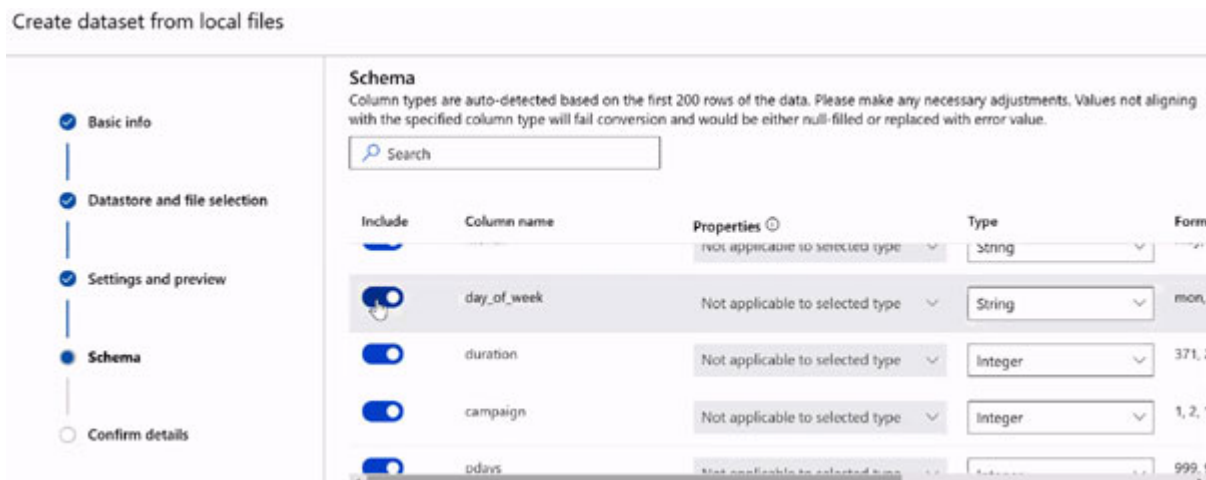
UI

Files will be uploaded to '\$(Upload path)/12-20-2021\_050636.UTC'

☐ Skip data validation

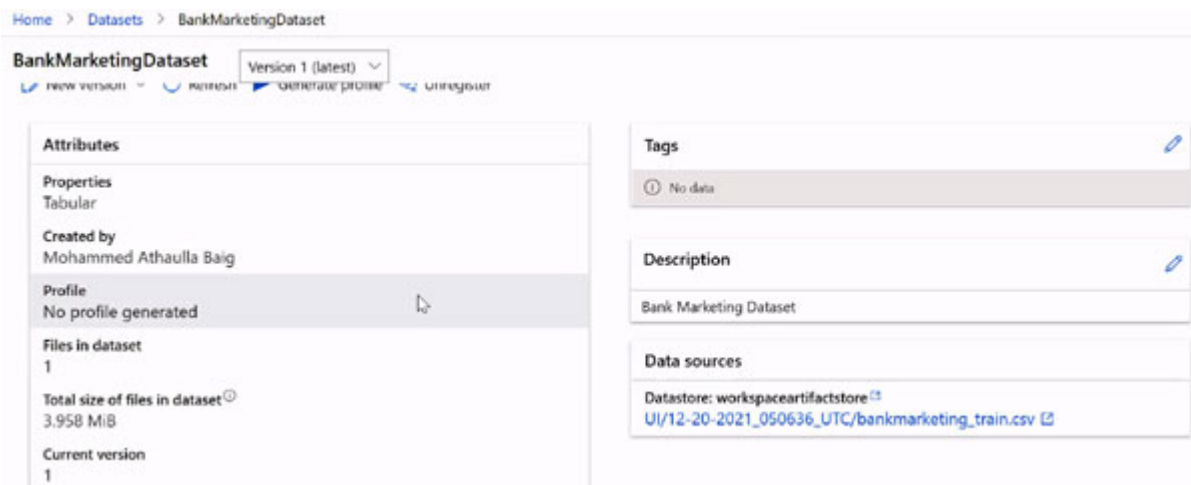
*Figure 8.6: Add Details*

Choose the columns not to include. Here we are choosing 'day of week'.



*Figure 8.7: Select schema*

The uploaded screen has the datasource mentioned as follows:



*Figure 8.8: Uploaded datasource screen*

### Step 3: Run the Experiment:

After we load and configure data, we can set up the experiment. This setup includes experiment design tasks such as, choosing the right computer instance size and choosing the column we want to use for prediction.

We will go to 'Create Automated ML' screen and choose the dataset we created as input:

Create a new Automated ML run

☒ Select dataset
   
  
☐ Configure run
   
  
☐ Select task and settings
   
  
☐ [Optional] Validate and test

### Select dataset

Select an input dataset from the list below, or create a new dataset. Automated ML currently only supports tabular data for authoring runs.

[+ Create dataset](#)
[Refresh](#)
☒ Show supported datasets only

All filters Clear all

Showing 1-4 of 4 datasets

Dataset name	Dataset type	Created on ↓	Modified on
BankMarketingDataset	Tabular	Dec 20, 2021 10:41 AM	Dec 20, 2021
nycdata	Tabular	Dec 16, 2021 10:26 AM	Dec 16, 2021

**Figure 8.9: Select Dataset**

We will go to Create a New Experiment or choose an existing experiment and configure run:

☒ Select dataset
   
  
☒ **Configure run**
  
  
☐ Select task and settings
   
  
☐ [Optional] Validate and test

### Configure run

☒ Select existing
 ☐ Create new

Existing experiment

Target column \*

Select compute type

Select Azure ML compute cluster \*

[+ New](#)
[Refresh computes](#)

**Figure 8.10: Configure Run**

We will configure the Compute Instance or cluster properties as shown below:



### Configure Settings

Configure compute cluster settings for your selected virtual machine size.

Name	Category	Cores	Available quota	RAM	Storage	Cost/Node
Standard_DS12_v2	Memory optimized	4	298 cores	28 GB	56 GB	\$0.30/hr

Compute name \* ⓘ

Minimum number of nodes \* ⓘ

*Figure 8.11: Configure Settings*

The following is the snapshot of the running experiment:

Home > Automated ML > mlexperiment > heroic\_square\_2bvtxfgd

heroic\_square\_2bvtxfgd

Refresh Cancel Delete

Details

Data guardrails

Models

Outputs + logs

Child runs

Snapshot

Properties

Status

Running

Setting up the run

Validating run configuration

Best model summary

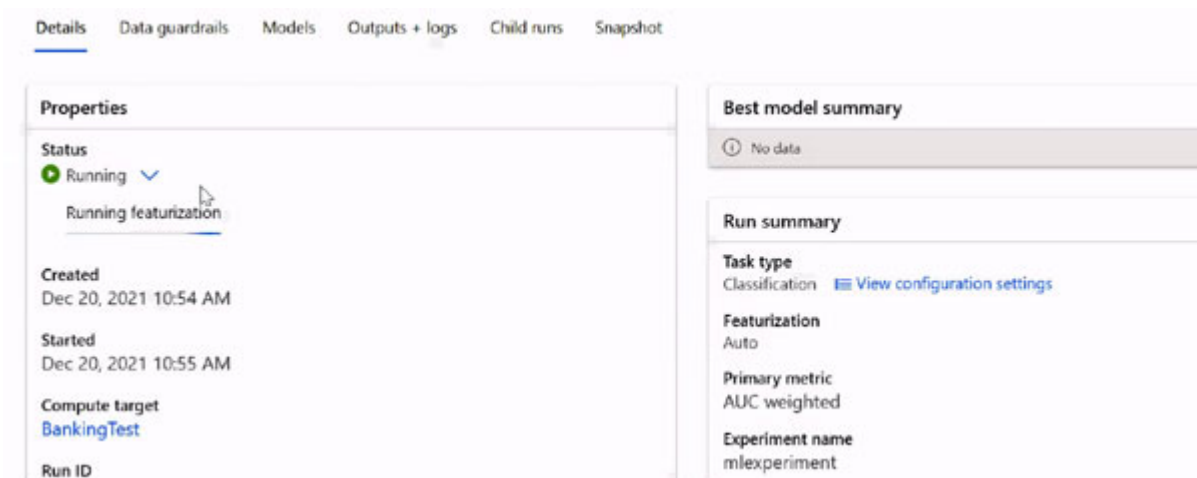
No data

Run summary

Task type

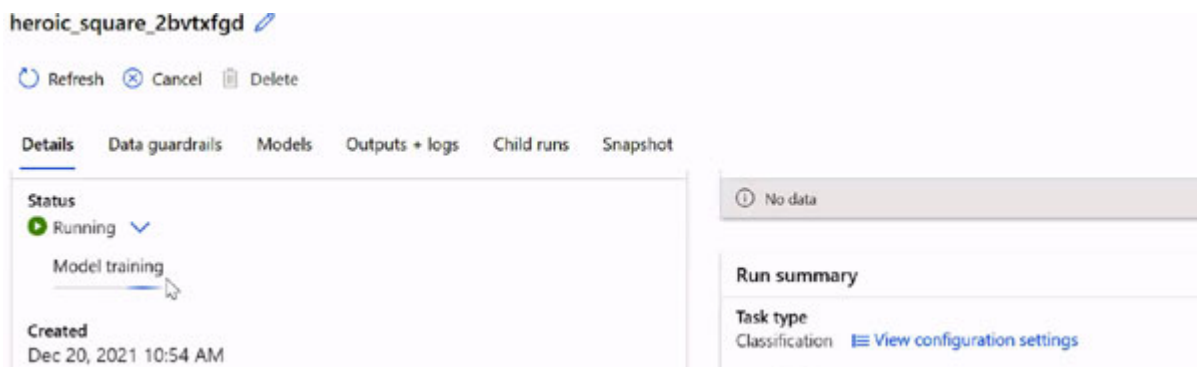
*Figure 8.12: Running experiment*

Following is the snapshot of the running experiment working on featurization:



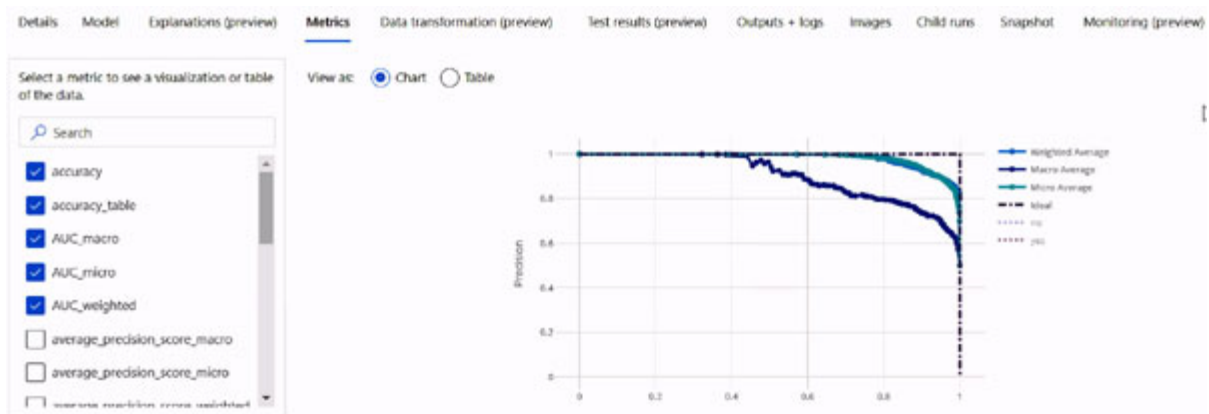
**Figure 8.13: Running experiment - Featurization**

Following is the snapshot of the running experiment training on the model:



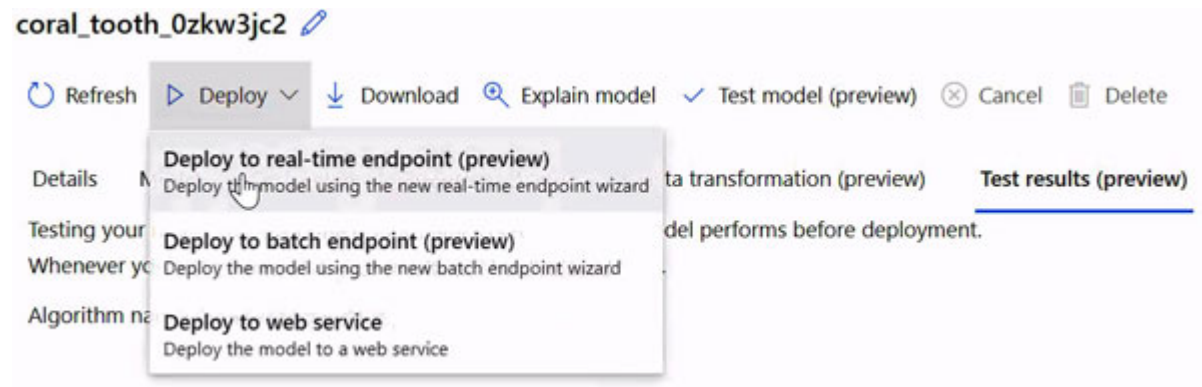
**Figure 8.14: Running experiment - Training on the model**

The results are displayed as follows:



**Figure 8.15: Results Window**

**Deploy the model:** We can deploy the model using the below shown options:



*Figure 8.16: Deploy the Model*

## [Sample Lab using Azure ML Studio Azure Machine Learning designer](#)

**Problem Statement:** We will run a sample application using Azure ML Studio Azure Machine Learning Designer. In this sample, we will leverage AutoMobile pricing data. We will import and clean the data. We will train, test and evaluate the model.

In this lab we will perform the following steps:

**Step 1:** Create a new pipeline.

**Step 2:** Import data.

**Step 3:** Prepare data.

**Step 4:** Train machine learning model using the prepared data.

**Step 5:** Evaluate the machine learning model.

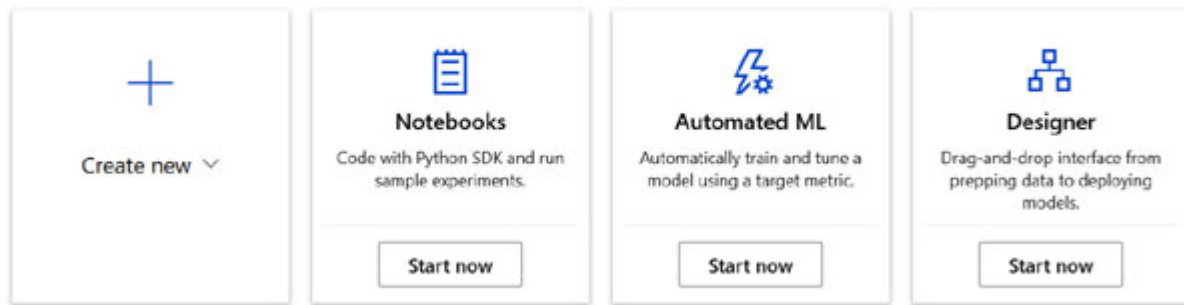
**Step 6:** Submit the pipeline.

Step 1: We will create a pipeline by navigating to Azure Machine Learning Studio.

<https://ml.azure.com/>

Following is Azure Machine Learning Studio opening screen:

## Welcome to the Azure Machine Learning Studio

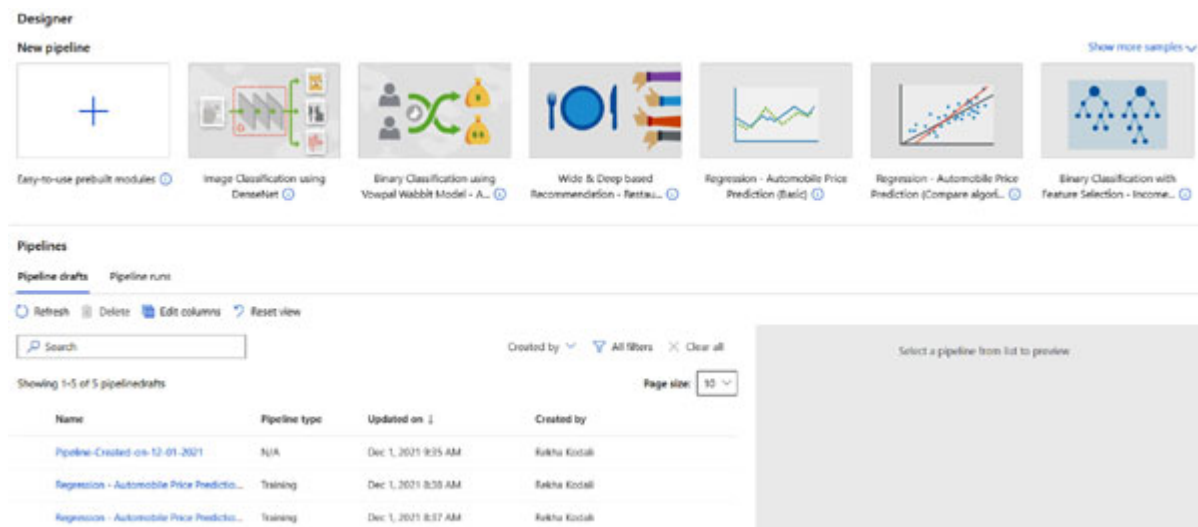


*Figure 8.17: Azure ML Studio*

Click on **Designer**.

This will bring up the below screen.

Given below is Create Pipeline screen:

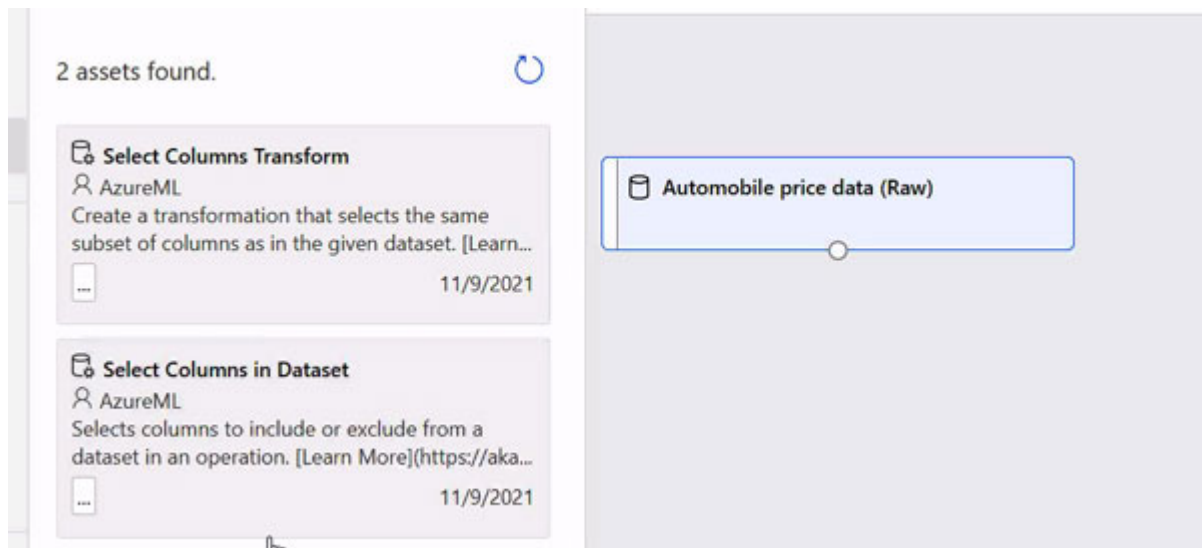


*Figure 8.18: Create Pipeline Screen*

Click on the first tab.

### Step 2: Import Data

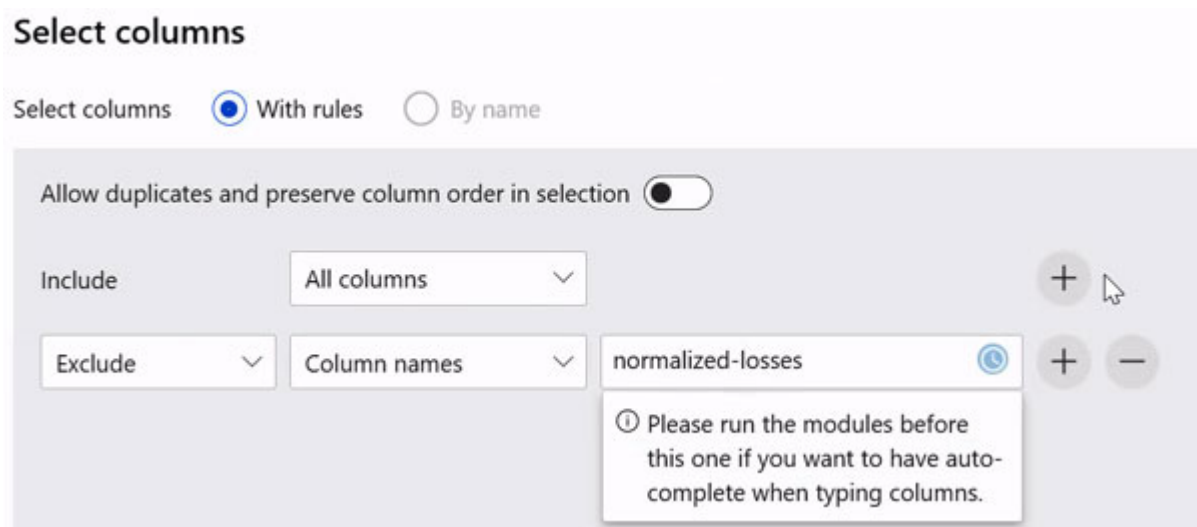
Add Dataset by choosing **Automobile price data (Raw)** in the sample dataset. You can see sample dataset on the left hand side. Choose Automobile price data.



*Figure 8.19: Import Data*

**Step 3: Prepare Data.** We need to clean missing rows and normalize the data before it can be used for the experiment.

Add Clean Missing Data and configure by clicking Edit Column as below:



*Figure 8.20: Select Columns*

Add clean missing data and configure by clicking **Edit Column** as follows:

### Columns to be cleaned

Select columns ☒ With rules ☐ By name

Allow duplicates and preserve column order in selection ☐

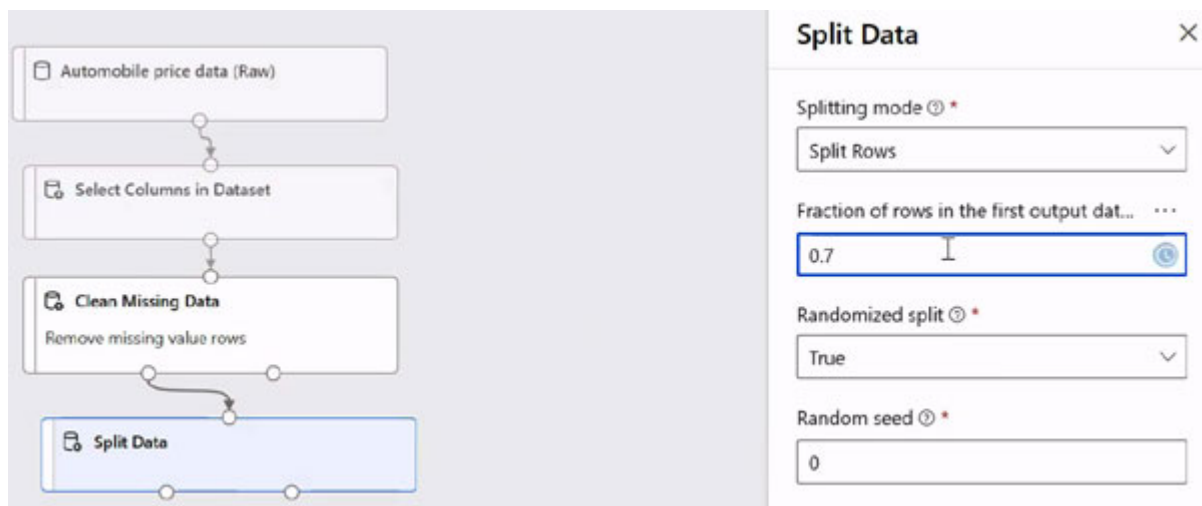
Include  +

*Figure 8.21: Clean Missing Data*

Choose ‘**Remove entire Row**’ in Cleaning Mode:

Add **Remove missing value rows** in the Comment section:

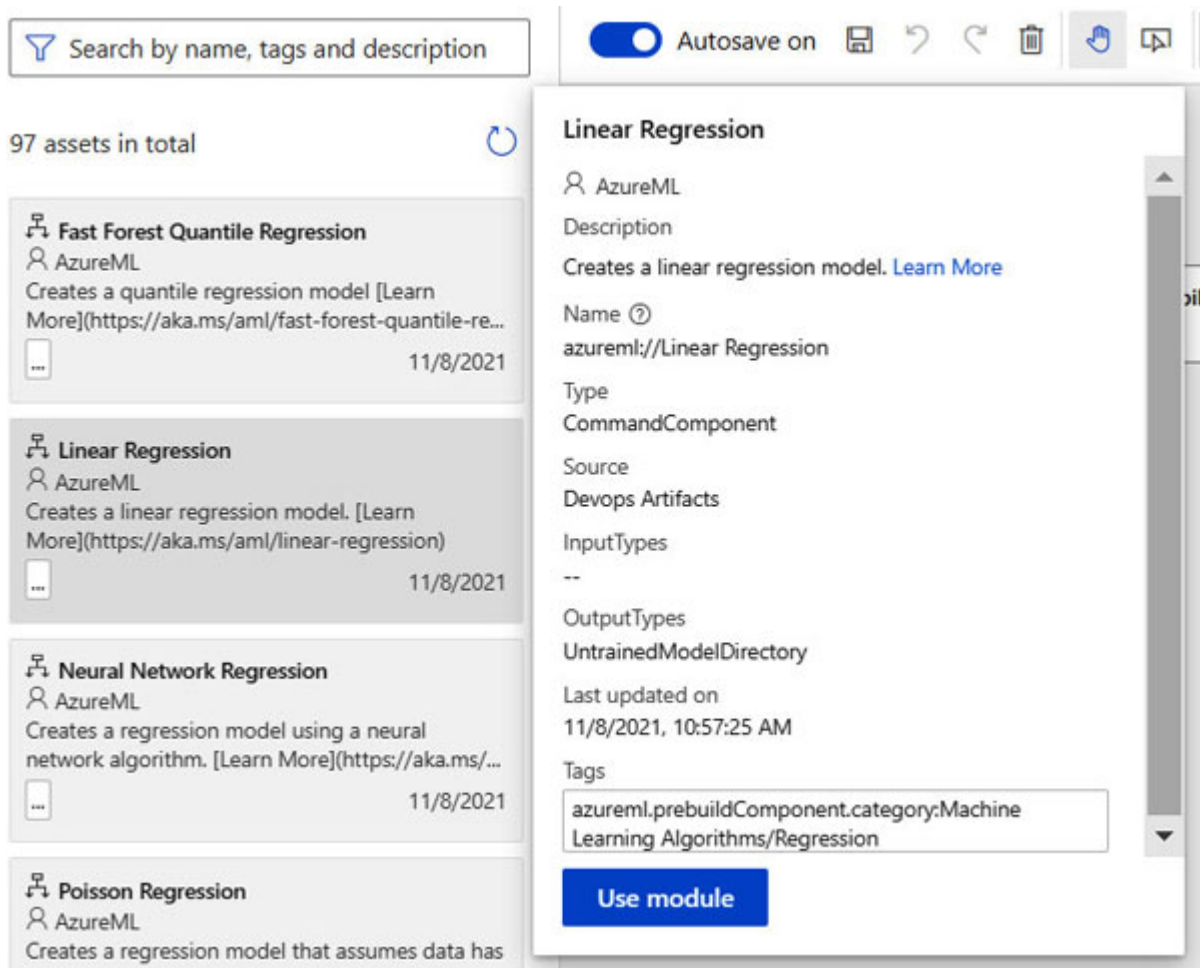
**Splitting the data:** For training, we will use ‘**Split Data**’ and adjust the fraction of rows percentage to .7. This uses 70% of the data for training and the remaining 30% for testing the model.



*Figure 8.22: Add Remove missing value rows in the Comment section*

**Step 4:** Train machine learning model. Now we will choose the algorithm to run the model.

Choose ‘**Linear Regression Model**’.



*Figure 8.23: Choose an algorithm*

## Add Train Model

In the Edit column selector, enter 'price' in the Label Column Dialog box.

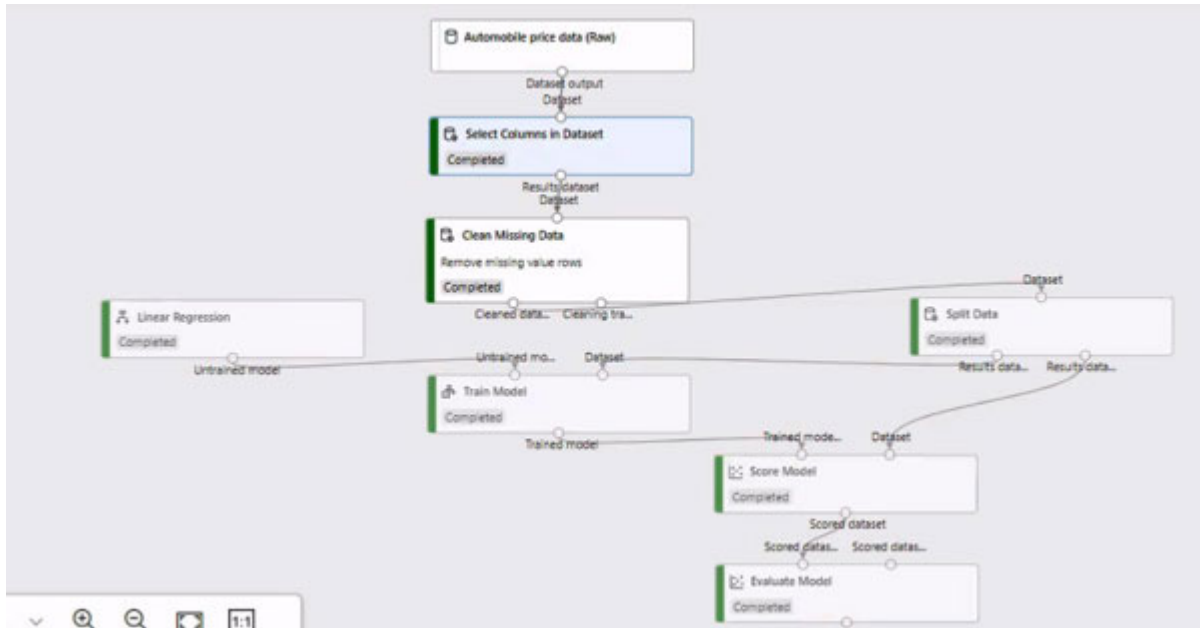


*Figure 8.24: Label the column*

## Step 5: Evaluate Model.

Add Score Model and Evaluate Model:

Complete the pipeline as shown below:



*Figure 8.25: The completed pipeline*

Choose and configure the Compute Instance settings:

Create compute instance

☒ Required Settings
   
☐ Advanced Settings

**Configure required settings**

Select the name and virtual machine size you would like to use for your compute instance. Please note that a compute instance can not be shared. It can only be used by a single assigned user. By default, it will be assigned to the creator and you can change this to a different user in the advanced settings section.

Compute name \* [?](#)

Location [?](#)

Virtual machine type [?](#)

☒ CPU ☐ GPU

Virtual machine size [?](#)

*Figure 8.26: Configure Settings*

We will configure the compute instance:



☒ Required Settings
   
☐ Advanced Settings

Name ↑	Category	Workload types	Av...	Cost
<input type="radio"/> Standard_DS11_v2 2 cores, 14GB RAM, 28GB storage	Memory optimized	Development on Notebooks (or other IDE) and light weight testing	296 c...	\$0.23/hr
<input checked="" type="radio"/> Standard_DS3_v2 4 cores, 14GB RAM, 28GB storage	General purpose	Classical ML model training, AutoML runs, pipeline runs (default compute)	296 c...	\$0.43/hr
<input type="radio"/> Standard_DS12_v2 4 cores, 28GB RAM, 56GB storage	Memory optimized	Training on large datasets (>1GB) parallel run steps, batch inferencing	296 c...	\$0.46/hr
<input type="radio"/> Standard_F4s_v2 4 cores, 8GB RAM, 32GB storage	Compute optimized	Real-time inferencing and other latency-sensitive tasks	296 c...	\$0.22/hr

Create

Back

Next: Advanced Settings

Cancel

*Figure 8.27: Configure the Compute Instance*

### Step 6: Submit the pipeline

We can submit the pipeline to train the machine learning model.

At the top of the canvas, select **Submit**.

We need to fill in the experiment details and run the pipeline

In the **Set up pipeline run** dialog box, select **Create new**.

**Set up pipeline run**

**Experiment**

☐ Select existing ☒ Create new

*Customers should not include personal data or other sensitive information in fields marked with the because the content in these fields may be logged and shared across Microsoft systems to facilitate operations and troubleshooting. [Learn more](#)*

New experiment name <sup>\*</sup>

mlexperi

Run description

Pipeline created on 20211211

☒ Continue on failure step

Compute target

**Submit** **Cancel**

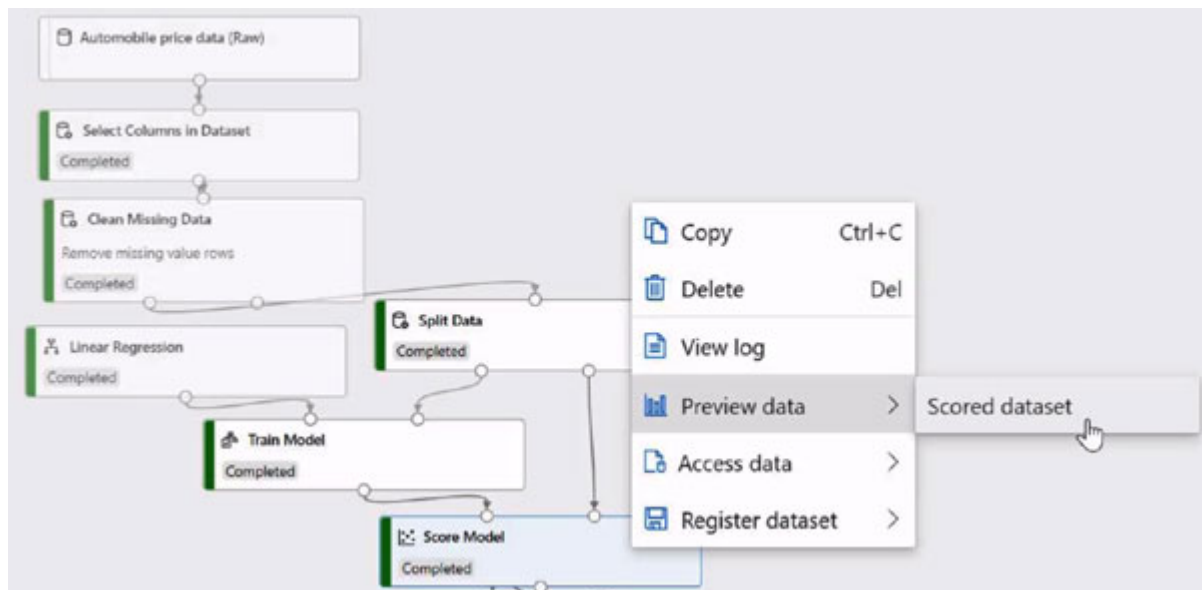
*Figure 8.28: Set up the pipeline run*

### Scoring the Model:

We can use the Score Model to apply our model to data.

- The inputs to the Score Model are a deployed model and a dataset of new, unlabelled records that we want to score.
- The output is a scored dataset showing predictions for each record.

The column '**Scored Labels**' predicts the prices for the automobiles based on the features we had selected. We can compare the predicted prices with the actual prices and ascertain the level of accuracy of our model.



*Figure 8.29: Click on Scored Dataset*

We can check the scored dataset. There is a new column of Scored Labels. This is the predicted data which is the automobile price.

The scored dataset looks like as follows:

Scored_dataset									
Rows 58 Columns 26									
symboling	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length
2	toyota	gas	std	two	hardtop	rwd	front	98.4	176.2
1	dodge	gas	std	two	hatchback	fwd	front	93.7	157.3
-1	volvo	gas	turbo	four	wagon	rwd	front	104.3	188.8
0	honda	gas	std	four	sedan	fwd	front	96.5	175.4
-1	toyota	gas	std	four	hatchback	fwd	front	102.4	175.6
-1	toyota	gas	std	four	hatchback	fwd	front	102.4	175.6
0	honda	gas	std	two	hatchback	fwd	front	96.5	167.5
0	toyota	gas	std	four	sedan	fwd	front	95.7	166.3
0	honda	gas	std	four	sedan	fwd	front	96.5	175.4
2	volkswagen	diesel	std	two	sedan	fwd	front	97.2	171.7

*Figure 8.30: Scored Dataset*

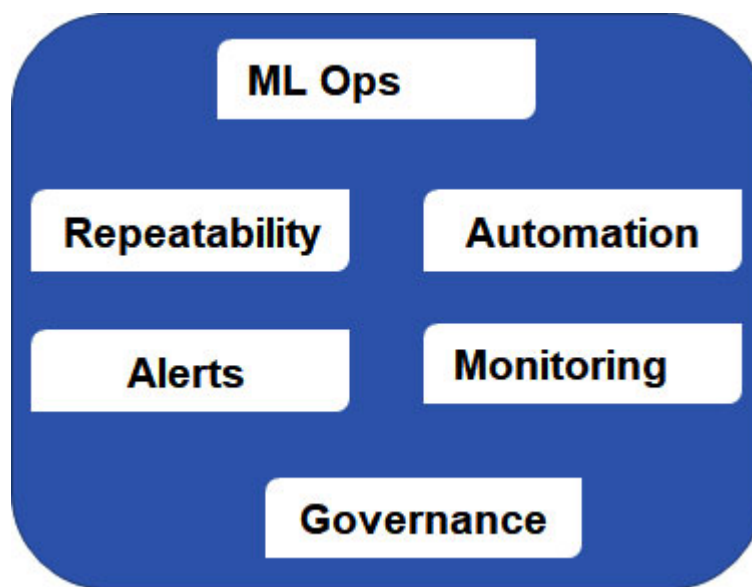
**Deploying the Model:** Once the model is deployed, it is available for consumption as Web-service and can be consumed via Azure Machine

Learning Test service or with API Testing tools. While deploying the model, we can set the checkbox to Enable the Application Insights diagnostics for data collection.

We have created two samples using ML Studio. We will now look at how we can migrate from the classic version to the newer version of ML Studio.

## ML Ops

**machine learning operations (ML Ops)** helps in developing, deploying models much faster and in automation of the end to end life cycle.



*Figure 8.31: MLOps*

### **Repeatability:**

**Create Pipelines:** Helps in creating pipelines for repeating steps like data preparing, scoring and training.

**Create Environments:** Helps in creating environments for training and deploying models.

**Automation:** Helps in automating the end-to-end workflow which includes updating models, training and deploying the models.

**Alerts:** Helps in creating alerts for model registration, experiment completion and model deployment.

**Monitoring:** Helps in comparing model inputs between training and inference, exploring model-specific metrics, and provides monitoring and

alerts on the machine learning infrastructure. It helps in monitoring operational and machine learning-related issues.

**Governance:** Helps in transparent sharing of the entire lifecycle from creation of models to deployment between data scientists, engineers, analysts and operations personnel.

For more information: MLOps with Azure Machine Learning - Cloud Adoption Framework | Microsoft Docs

## Migration

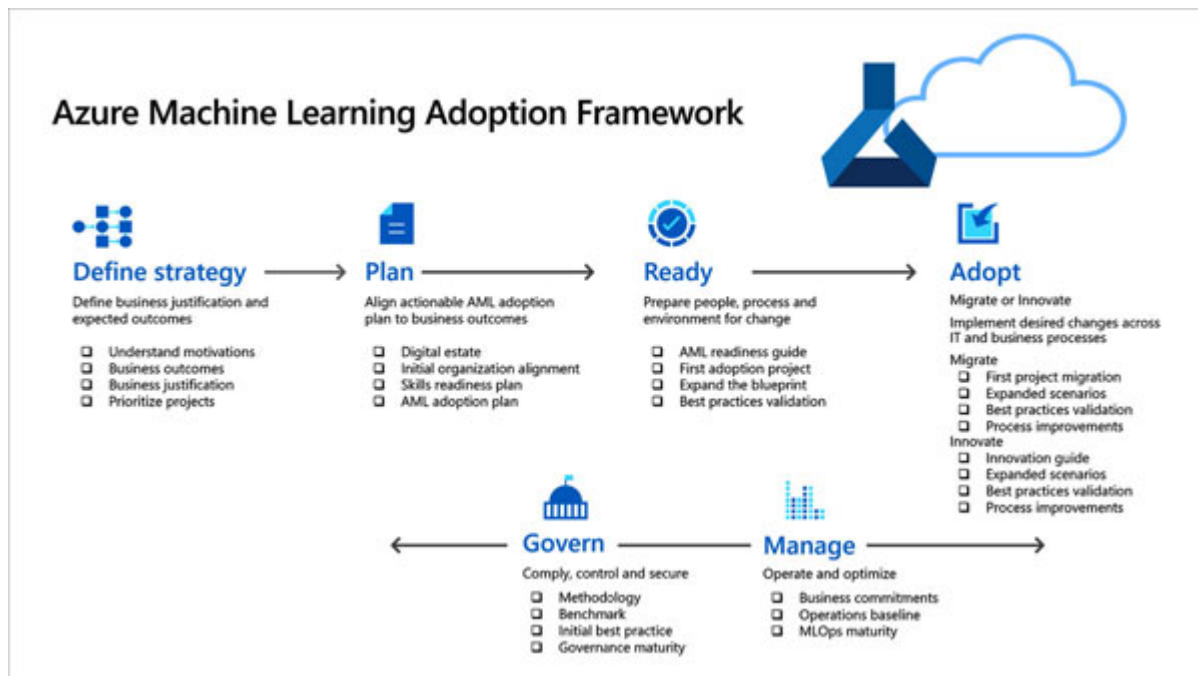
ML Studio (classic) will be discontinued from August 2024. We will not be able to create new Machine Learning Studio (classic) resources from December 2021.

It is recommended to transition to the new studio by 31 August 2024 as support for Machine Learning Studio (classic) will end by then.

The migration includes assessing the current state, planning, getting ready, adopting new features, and governing and managing assets and resources.

Given below is Azure Machine Learning Adoption Framework which we can use as a guideline for migration:

Reference: <https://docs.microsoft.com/en-us/azure/machine-learning/migrate-overview>



*Figure 8.32: Azure Machine Learning Adoption Framework*

**Azure/Azure-Machine-Learning-Adoption-Framework:** This repo provides best practice guidance, plan template, solution assessment tool etc. to help Machine Learning Studio(classic) customer adopt Azure Machine Learning. ([github.com](https://github.com/Azure/Azure-Machine-Learning-Adoption-Framework)).

While we have seen some sample programs using Azure Machine Learning Studio, Azure Machine Learning supports a lot of algorithms. We will explore the details of the same in the next section.

## [Azure ML library of algorithms](#)

Azure Machine Learning has a large library of algorithms which help solve regression, classification, recommender systems, anomaly detection, clustering, etc. The Azure Machine Learning Algorithm Cheat Sheet helps us identify the appropriate algorithm from the designer.

We can download the cheat sheet here: <https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet>

Here is a view of the sheet from the Microsoft site:

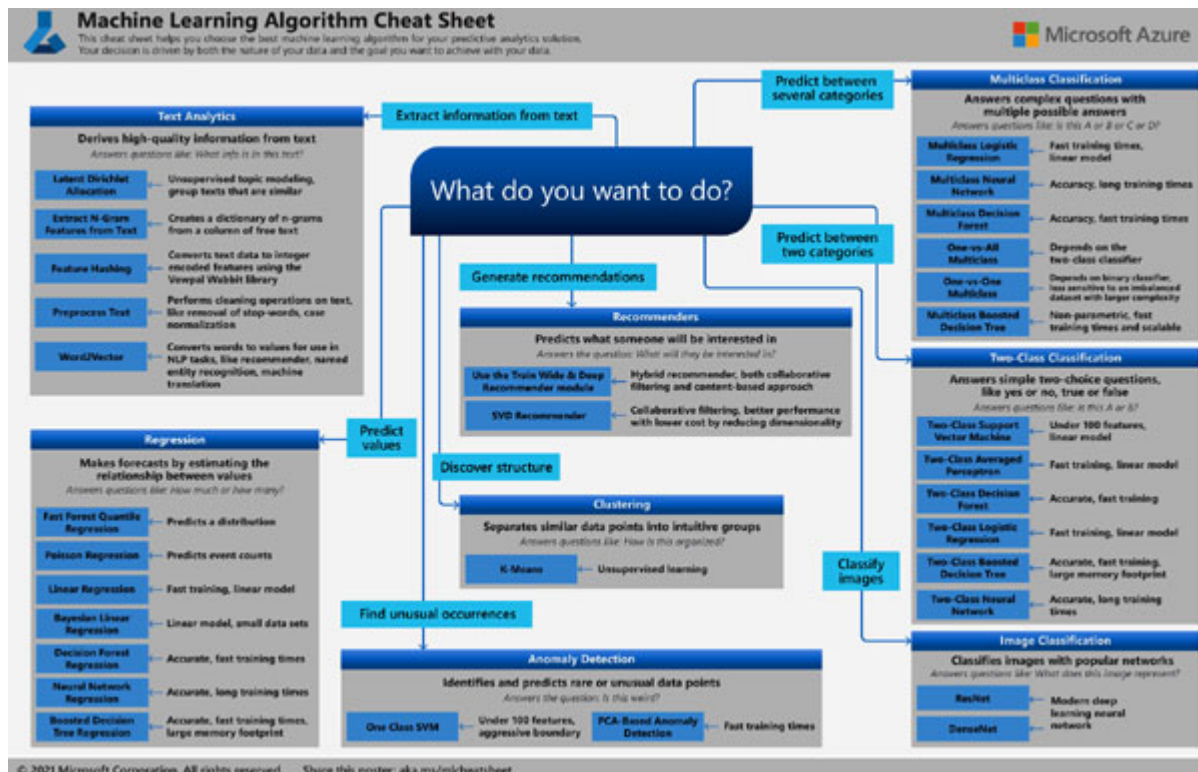


Figure 8.33: ML algorithm Cheat Sheet

**Using the Machine Learning algorithm cheat sheet:** We can use this sheet as a starting reference point. To solve a particular problem, we might have to use a combination of several algorithms.

We can learn more about the algorithms supported by Azure ML Designer at

Algorithm and component reference. (<https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/component-reference>)

While we have discussed Microsoft based libraries we will briefly touch upon the Accord.NET framework which provides a variety of libraries for image processing apart from other tasks.

## **Accord.NET -A brief Introduction**

Accord.NET framework comprises a set of libraries for statistics, numerical optimization, machine learning, artificial neural networks, numerical linear algebra, signal and image processing, and support libraries.

It can be used in Microsoft Windows, Xamarin, Unity3D, Windows Store applications, Linux or mobile applications.

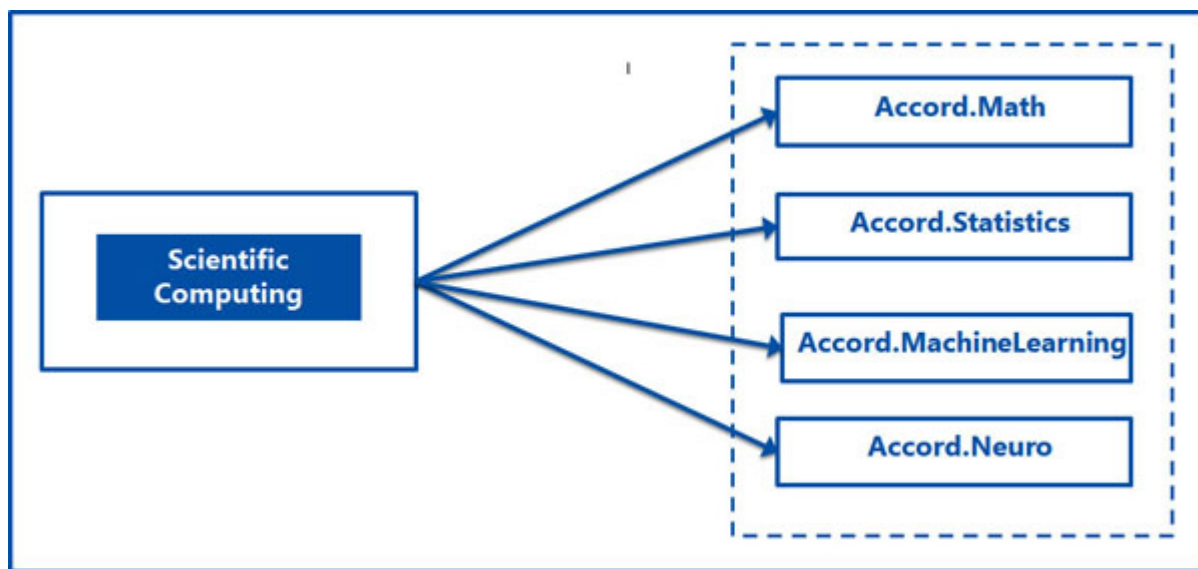


The framework offers a consolidated API for learning/training machine learning models that is both easy to use and extensible.

The framework has been divided into modules and users can choose the modules they would like to include in their projects.

Algorithms supported by machine learning can be classified as supervised and unsupervised learning.

Given below are the some of the algorithms supported by Accord.NET across various modules:



*Figure 8.34: Accord.NET Framework*

## Conclusion

In this chapter, we have looked at Azure Machine Learning Studio. We have learned how to create sample ML experiments using Azure Machine Learning Studio.

## Multiple choice questions

This section consists of a set of questions that are designed to test your knowledge of the concepts covered in this chapter.

### **1. What is a project authoring and asset management portal?**

- a. Azure Machine Learning Studio



- b. VS code
  - c. Visual Studio
  - d. Eclipse
2. **What provides a no code based approach which helps in developing ML applications using simple drag and drop without writing any code?**
- a. Automated ML
  - b. ML Analytics
  - c. Visual Studio
  - d. Eclipse
3. **What helps in developing, deploying models much faster and in automation of the end to end life cycle?**
- a. DevOps
  - b. AIML
  - c. MLOps
  - d. All
4. **What is the foundational block for Azure machine learning project?**
- a. Dataset
  - b. Project
  - c. Experiment
  - d. Machine Learning Workspace
5. **What helps in transparent sharing of the entire lifecycle from creation of models to deployment between data scientists, engineers, analysts and operations personnel?**
- a. Automation
  - b. Governance
  - c. Retry
  - d. Transparency

## **Answers**

- a. **a. Azure Machine Learning Studio**
- b. **a. Automated ML**
- c. **c. MLOps**
- d. **d. Machine Learning Workspace**
- e. **b. Governance**

# Index

## A

Accord.NET framework [227](#), [228](#)

AI services [4](#)

    conversational AI [6](#)

    custom services [7](#), [8](#)

    Trained Services (Prebuilt) [5](#)

algorithms, ML.NET [174](#)

    selecting [174](#)

    semi-supervised learning [175](#)

    supervised learning [174](#)

    unsupervised learning [175](#)

    view [176](#)

Analyze Image API [13](#)

    features [14](#)

    leveraging, for image analytics [26](#)

Anomaly detection

    sample use case flow [72](#), [73](#)

    use cases [72](#)

Anomaly detector [72](#)

APIs, Azure Vision Service

    accessing [17](#)

    Analyze Images API [13](#), [14](#)

    Computer Vision [11](#), [12](#)

    Custom Vision Service [15](#)

    Face API [16](#)

    Spatial Analysis [15](#)

    Text Extraction [12](#)

    use cases [18](#), [19](#)

Artificial Intelligence [124](#)

Audio Insights

    features [104](#), [105](#)

automated machine learning UI [208](#)

Azure AI platform [2](#)

Azure Applied AI Services [2](#), [97](#)

    AI Infrastructure [4](#)

    Azure Bot Service [3](#)

    Azure Cognitive Search [3](#)

    Azure Cognitive Services [4](#)

    Azure Form Recognizer [3](#), [98](#)

    Azure Immersive Reader [3](#), [98](#)

    Azure Machine Learning [4](#)

    Azure Metrics Advisor [3](#), [98](#)

    Azure Video Analyser for Media [3](#)

- overview [98](#)
- Azure Bot Service [136](#)
- Azure Cognitive Search [108](#), [109](#)
  - approaches [109](#), [110](#)
  - cognitive skills, adding [113](#)
  - cognitive skills, attaching [114](#), [115](#)
  - components [109](#)
  - creating [110](#), [111](#)
  - enrichments, adding [114](#)
  - enrichments, saving to Knowledge Store [115](#)
  - index, creating [112](#)
  - indexer, configuring [116](#)
  - progress, monitoring [117](#)
  - Search explorer, using [117](#), [118](#)
- Azure Cognitive Services [8](#), [71](#)
  - Decision Services [9](#), [72](#)
  - Docker containers [92](#)
  - Language Service [9](#), [76](#)
  - Speech Service [10](#), [88](#), [89](#)
  - Vision Service [10](#)
  - Web Search [11](#), [90-92](#)
- Azure Cognitive Services for Language [133](#), [134](#)
- Azure Form Recognizer [98](#), [106](#)
  - accessing [107](#), [108](#)
  - Azure Bot Service [98](#)
  - Azure Cognitive Search [98](#)
  - Azure Video Analyzer for Media [98](#)
  - custom models [106](#)
  - prebuilt models [106](#)
  - supported models [106](#)
- Azure Functions [81](#)
  - triggers [82](#)
- Azure Immersive Reader [98](#), [101](#)
  - features [101](#), [102](#)
- Azure Logic Apps [80](#)
  - characteristics [81](#)
  - templates [81](#)
- Azure Machine Learning
  - library of algorithms [226](#)
- Azure Machine Learning Adoption Framework [225](#), [226](#)
- Azure Machine Learning Algorithm Cheat Sheet [226](#), [227](#)
- Azure Machine Learning Designer [208](#)
  - algorithms [227](#)
- Azure Metrics Advisor [98](#)
  - features [99](#)
  - flow diagram [100](#)
  - recommendations [100](#)
  - use cases [99](#)
  - workflow, creating [101](#)
- Azure ML Studio [207](#), [208](#)

- author machine learning projects [208](#)
- sample lab, with automated machine learning UI no-code approach [208-216](#)
- sample lab, with Azure Machine Learning designer [216-224](#)
- Azure Video Analyzer for Media [102](#)
  - use cases [102](#), [103](#)
  - working [103](#)
- Azure Vision APIs, use cases
  - Content Moderator [20](#)
  - Emotional Analysis [19](#), [20](#)
  - Smart Image Classification and Object Detection [18](#), [19](#)
  - Smart Security [19](#)
- Azure Vision Service [11](#)

## B

- Bing web search APIs
  - using [90-92](#)
- Bot builder tools [136](#)
- Bot ecosystem [125](#)
  - Bot Framework [126](#)
  - Bot Framework Composer [128](#)
  - Cognitive Intelligence [132](#)
  - informational sources [136](#)
  - Search [135](#)
  - Storage [135](#)
- Bot Framework [126](#)
  - conversation experiences [136](#)
  - key considerations [126](#), [127](#)
- Bot Framework Composer
  - Actions [130](#)
  - building [165](#), [166](#)
  - configuration [129](#)
  - customer scenario [167](#)
  - dialogs [129](#)
  - enterprise scenarios [166](#), [167](#)
  - features [129](#)
  - memory scopes [131](#)
  - Natural Language Processing (NLP) [130](#)
  - recognizers [129](#), [130](#)
  - setting up [128](#), [129](#)
  - skills [131](#)
  - tasks, performing [128](#)
  - Triggers [130](#)
- Bot Framework Composer, use cases [167](#)
  - Natural Language Processing [168](#), [169](#)
  - re-engineer operational processes [168](#)
  - re-imagine enterprise models [168](#)
  - re-interpret customer interactions [168](#)
- Bot Framework Emulator [139](#)
  - basic bot, building [140](#)

- BasicBot source code, updating [161](#)
- chat files [152](#)
- code, building [164](#)
- code testing [164](#)
- dispatch model, creating with Bot Framework Orchestrator [156-160](#)
- download link [139](#)
- features [151](#)
- LUIS, adding to sample [155](#)
- LUIS Model, adding [155](#)
- Oliver Garden FAQ, adding [165](#)
- QnA KB, creating [140-148](#)
- second QnA KB, adding [156](#)
- transcripts [151](#)
- using [148-155](#)
- Bot Framework SDK [136](#)
- Bot functionality [6](#)
  - features [6](#)
  - leveraging [6](#)
- Bots [124, 125](#)
  - building, with Bot Framework [136](#)
  - lifecycle [137](#)
- BotServices [161](#)
- Botting [127](#)
- bot transcript file [151](#)

## C

- Chatdown
  - installing [153](#)
- chat file [152](#)
- Cognitive Intelligence [132](#)
  - Azure Cognitive Services for Language [133, 134](#)
  - LUIS [132](#)
  - QnA Maker (Managed) [132](#)
- Cognitive Toolkit (CNTK) [174](#)
- Computer Vision API [11](#)
- Content Moderator [73](#)
  - detailed APIs [75](#)
  - use case [74](#)
- conversational AI
  - use cases [7](#)
- Conversational User Interfaces (CUI) [125](#)
- Conversation as a Service (CaaS) [125](#)
- custom services [7](#)
  - use cases [7, 8](#)
- custom vision models
  - training [15, 16](#)
- Custom Vision Service API [15, 16](#)
  - image classification, performing with [36](#)
  - object detection, performing with [41](#)

Prediction API [16](#)  
pricing [16](#)  
Training API [16](#)

## D

Data API, ML.NET  
  data transforms [173](#)  
  IDataView [173](#)  
Decision Services [9](#)  
  Anomaly detector [9](#), [72](#)  
  Content Moderator [9](#), [73](#)  
  Personalizer Service [9](#), [75](#)  
decision tree trainers [177](#)  
  Fast Tree [178](#)  
  Generalized Additive Model [178](#)  
  Light Gradient Boosted Machine [178](#)  
Developer Friendly APIs, ML.NET  
  Data [173](#)  
  Extensions [174](#)  
  Model Consumption and Evaluation [174](#)  
  Model Training [173](#)  
  Tools [174](#)  
developer tools  
  setting up [20](#)

## E

Entity Recognition API [79](#)  
ErrorHandlerAdaptor [162](#)

## F

Face API [17](#), [55](#)  
  Face Detection [16](#), [17](#)  
  Face Recognition [16](#), [17](#)  
Face API, for Face Recognition  
  Emotion Recognition [56](#)  
  Face Grouping [56](#)  
  use cases [55](#), [56](#)  
Face API lab  
  sample project, creating [57-64](#)  
Face Recognition [54](#), [55](#)  
  Face Grouping [17](#)  
  Face Identification [17](#)  
  Face Verification [17](#)  
function bot [136](#)

## H

handwriting recognition lab  
sample application, creating [48-54](#)

## I

IBotService.cs [161](#)  
image analytics, with Analyze Image API  
area of interest [28](#)  
brands, detecting [27](#), [28](#)  
categorizations [27](#)  
color scheme detection [28](#)  
domain-specific models, listing [28](#)  
face detection [28](#)  
image, describing [28](#)  
image type detection [28](#)  
lab [28](#)  
object detection [26](#), [27](#)  
performing [26](#)  
sample project, creating [28-36](#)  
image classification, with Azure Custom Vision Service  
lab [36](#)  
performing [36](#)  
sample project, creating [36-41](#)  
image processing [26](#)  
Immersive Reader [76](#)

## K

Key Phrase Extraction API [79](#)  
key pillars, reference architecture  
Azure Bot Service [138](#), [139](#)  
Conversational Channels [139](#)  
security [139](#)  
users [138](#)  
Knowledge Store [115](#)

## L

Language Detection API [79](#)  
Language Service [9](#)  
Immersive Reader [76](#)  
LUIS [76](#)  
QnAMaker [76](#)  
Translator [76](#)  
Language Understanding Service (LUIS) [6](#), [76](#), [77](#), [132](#)  
lifecycle, Bot  
build [137](#)  
connect [138](#)  
plan [137](#)  
publish [138](#)



- refine [138](#)
- test [137](#)
- linear trainers [176](#)
  - averaged perceptron [177](#)
  - L-BFGS [177](#)
  - stochastic dual coordinated ascent [177](#)
  - symbolic stochastic gradient descent [177](#)
- logic app
  - creating, with OCR, Text Analytics and Azure Functions [82-88](#)
- LUIS app
  - components [133](#)
  - concepts [133](#)
  - creating [133](#)

## M

- ML.NET [171](#), [172](#)
  - algorithms [174](#)
  - algorithm, selecting [174](#), [176](#)
  - components [172](#), [173](#)
  - Developer Friendly APIs [173](#)
  - sample application, creating for fare prediction [188-192](#)
  - sample application, creating for issue classification [192-197](#)
  - sample application, creating for sentiment analysis [183-187](#)
  - samples [182](#)
  - trainers [176](#)
- ML.NET model
  - creating [178-181](#)
  - using [182](#)
- ML.NET Model Builder [197](#)
  - data [198](#)
  - data training [198](#), [199](#)
  - sample application, creating for sentiment analysis [199-203](#)
  - templates [198](#)
- ML Ops [224](#), [225](#)

## N

- Natural Language Generation (NLG) [130](#)
- Natural Language Processing (NLP) [130](#)
  - language generation [130](#)
  - language understanding [130](#)
  - orchestration [131](#)
  - QnA [131](#)
- Natural Language Processing (NLP) service [76](#)
- Natural Language Understanding (NLU) [130](#)

## O

- object detection, with Azure Custom Vision Service

performing [41](#)  
OCR API  
features [47](#)  
use cases [47](#), [48](#)

## P

Personalizer Service [75](#)  
use cases [76](#)

## Q

QnAMaker [76](#)  
QnA Maker (Managed) [132](#)

## R

reference architecture [138](#)  
key pillars [138](#)  
Region of Interest (ROI) [28](#)  
Robotic Process Automation (RPA) [126](#)

## S

sample application, for fare prediction  
creating, with ML.NET [188-192](#)  
sample application, for issue classification  
creating, with ML.NET [192-197](#)  
sample application, for sentiment analysis  
creating, with ML.NET Model Builder [199-203](#)  
sample application, for sentiment analysis  
creating, with ML.NET [183-187](#)  
SdcaRegressionTrainer [176](#)  
Search, Bots ecosystem [135](#)  
Sentiment Analysis API  
using [79](#)  
skill [131](#)  
skillset [114](#)  
Spatial Analysis [15](#), [65](#)  
use cases [15](#), [66](#)  
Speech Service [88](#)  
leveraging [10](#)  
Speaker Recognition [89](#)  
Speech to Text [89](#)  
Speech Translation [89](#)  
Text to Speech [89](#)  
use cases [10](#)  
Voice Assistants [89](#)  
Storage, Bots ecosystem [135](#)  
supervised learning algorithm

classification [175](#)  
regression [175](#)

## T

Text Analytics API [77](#), [78](#)  
    Entity Recognition API [79](#)  
    Key Phrase Extraction API [79](#)  
    Language Detection API [79](#)  
    Sentiment Analysis API [79](#)  
    use case [80](#)  
Text Extraction API [12](#)  
    OCR API [13](#)  
    Read API [12](#)  
text mining search and navigation (TMSN) [171](#)  
text recognition  
    handwritten text recognition [46](#)  
    performing, with OCR API [46-48](#)  
    printed text recognition [46](#)  
Trained Services (Prebuilt), use cases  
    HealthCare [5](#)  
    Insurance [5](#)  
    Oil and Gas companies [5](#)  
trainers, ML.NET  
    decision tree trainers [177](#)  
    linear trainers [176](#)  
Translator [76](#)

## U

unsupervised learning algorithm [175](#)  
    association [175](#)  
    clustering [175](#)

## V

Video and Audio Insights  
    features [105](#)  
Video/Audio Indexer  
    features [104](#)  
Vision Service  
    use cases [10](#)  
VS Code [20](#)

## W

Web App Bot [136](#)  
Web Chat [127](#)  
Web Search [11](#)