



KodeKloud

© Copyright KodeKloud

Visit www.kodekloud.com to learn more.

Fundamental AI Concepts

Module Introduction

01 What is Artificial Intelligence (AI)?

02 Common AI Workloads



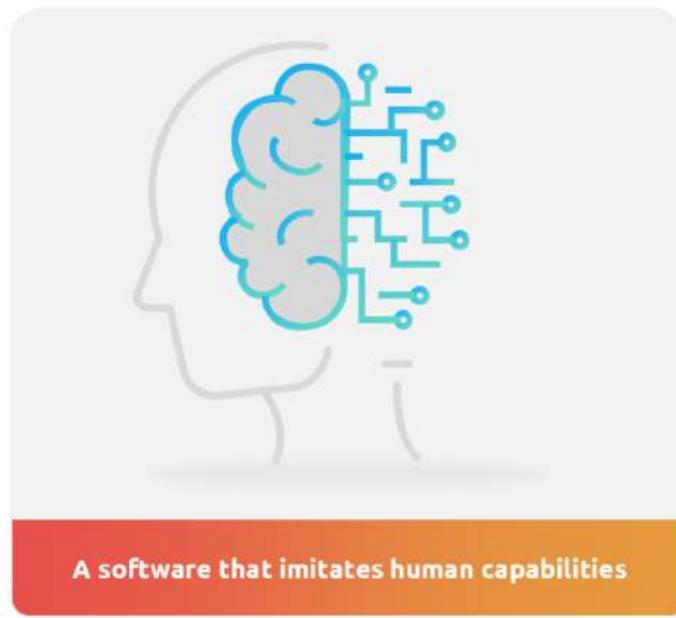
KodeKloud

03 Principles of Responsible AI

What is Artificial Intelligence?



What is Artificial Intelligence (AI)?



KodeKloud

A software that imitates human capabilities

Artificial Intelligence – Applications



01

Prediction and Pattern Recognition

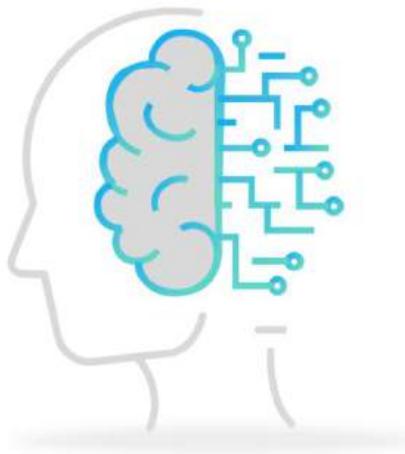


It is used in various applications such as forecasting sales, predicting maintenance needs, and identifying trends in consumer behavior.

© Copyright KodeKloud

Prediction and Pattern Recognition : AI can predict outcomes and identify patterns by analyzing historical data. This is used in various applications such as forecasting sales, predicting maintenance needs, and identifying trends in consumer behavior.

Artificial Intelligence – Applications

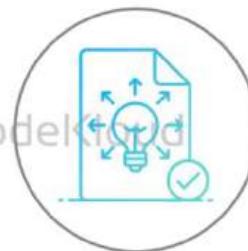


02

Anomaly Detection and Decision Making



AI recognizes
abnormal events



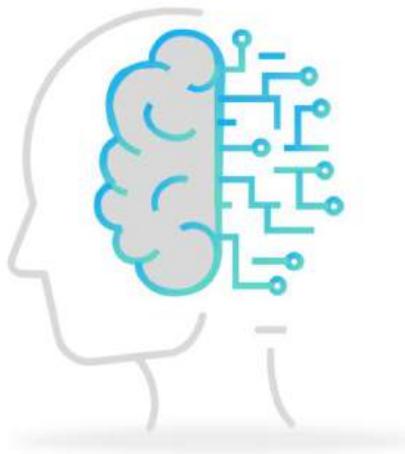
Appropriate
decision making

Example: In cybersecurity, AI can detect unusual activities that might indicate a security breach and take action to mitigate risks.

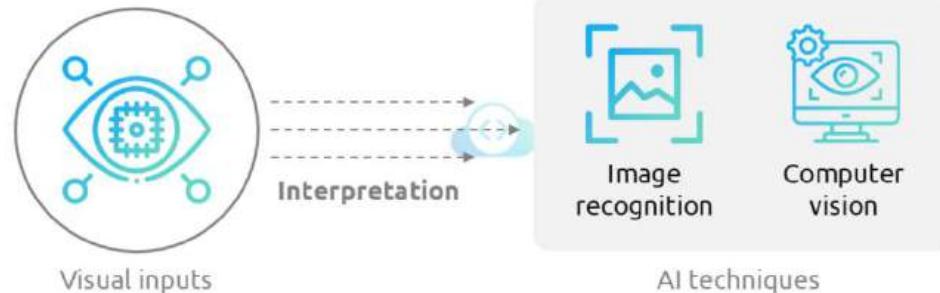
© Copyright KodeKloud

Anomaly Detection and Decision Making: AI systems can recognize abnormal events and make decisions accordingly. For example, in cybersecurity, AI can detect unusual activities that might indicate a security breach and take action to mitigate risks.

Artificial Intelligence – Applications



03 Visual Interpretation

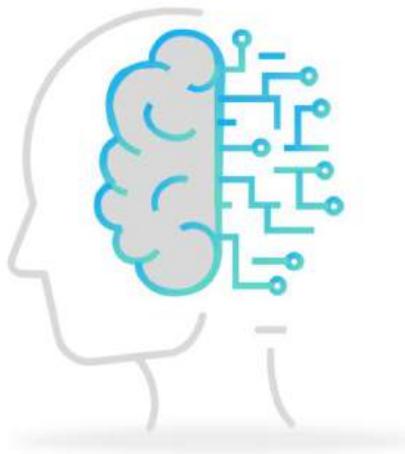


This technology is applied in areas such as medical imaging, autonomous driving, and facial recognition.

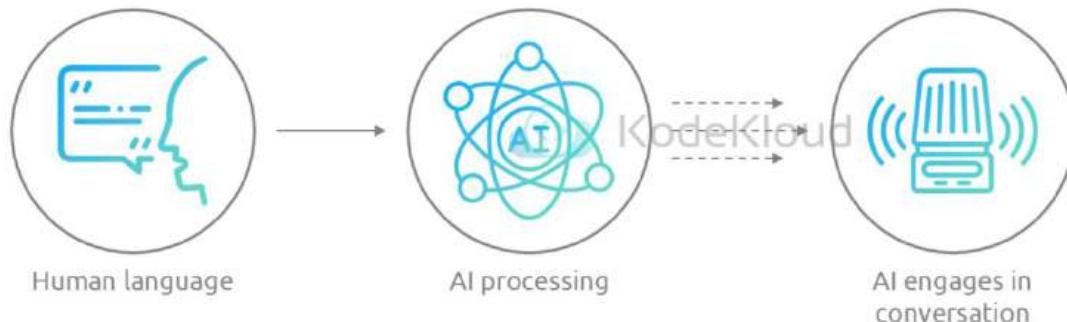
© Copyright KodeKloud

Visual Interpretation: AI is capable of interpreting visual inputs through techniques like image recognition and computer vision. This technology is applied in areas such as medical imaging, autonomous driving, and facial recognition.

Artificial Intelligence – Applications



04 Understanding Language and Conversation



Example: Virtual assistants like Siri and Alexa, which can comprehend spoken commands and respond appropriately

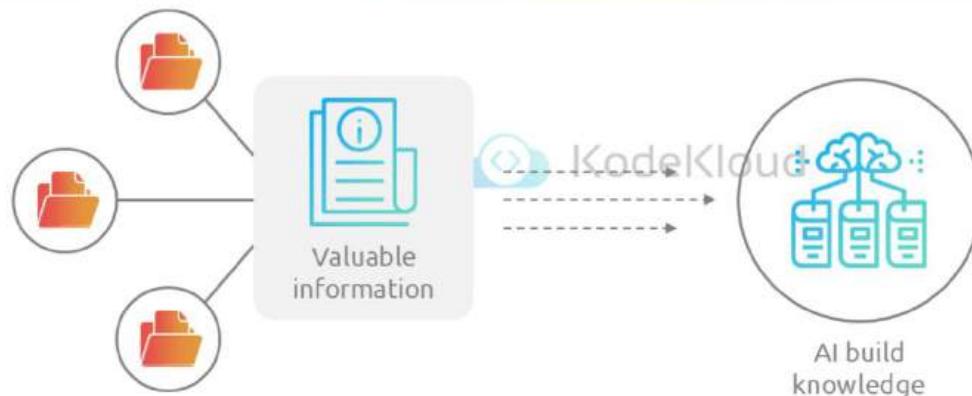
© Copyright KodeKloud

Language Understanding and Conversation: AI can understand and process human language, enabling it to engage in conversations. This is exemplified by virtual assistants like Siri and Alexa, which can comprehend spoken commands and respond appropriately

Artificial Intelligence – Applications



05 Information Extraction

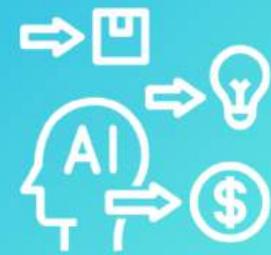


Example: Gather insights and assist in knowledge discovery from research papers, news articles, and other data sources.

© Copyright KodeKloud

Information Extraction: AI can extract valuable information from various sources to build knowledge. For instance, it can sift through research papers, news articles, and other data sources to gather insights and assist in knowledge discovery.

Common AI Workloads

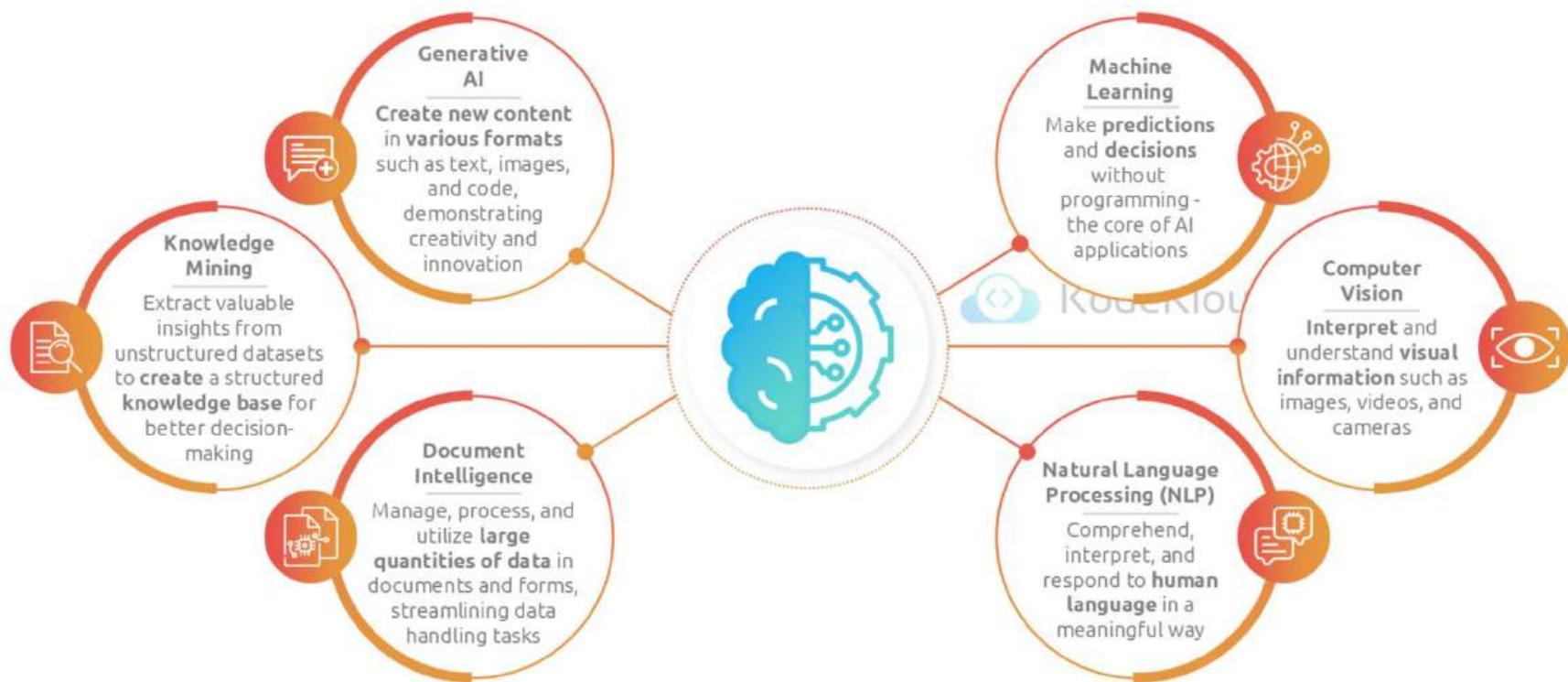


Common AI Workloads



KodeKloud

Common AI Workloads



© Copyright KodeKloud

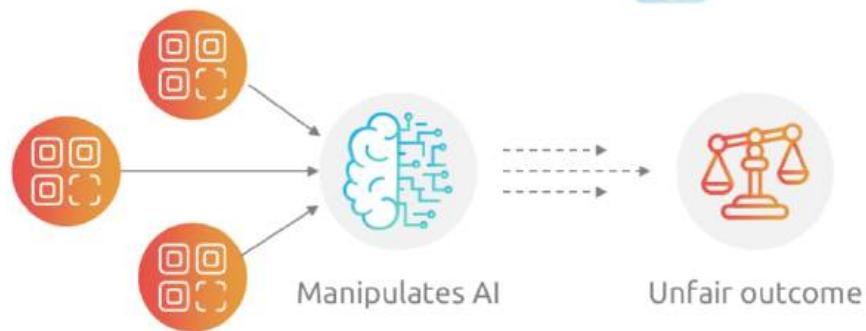
Machine Learning :AI models that learn from data to make predictions and decisions without explicit programming. It forms the core of various AI applications

Principles of Responsible AI



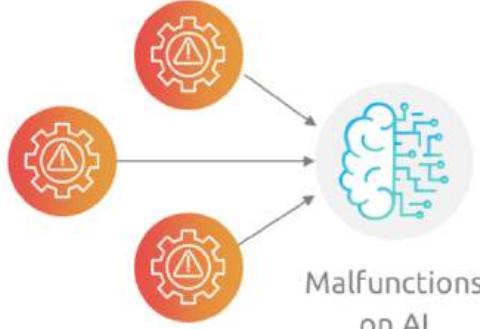
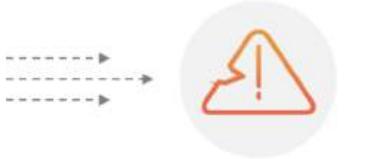
Principles of Responsible AI

Principle	Challenge	Example
Fairness 	AI systems can perpetuate existing biases or create new ones, leading to unfair outcomes	An AI system used for hiring recommendations might favor candidates of a certain gender or ethnicity due to biased training data 



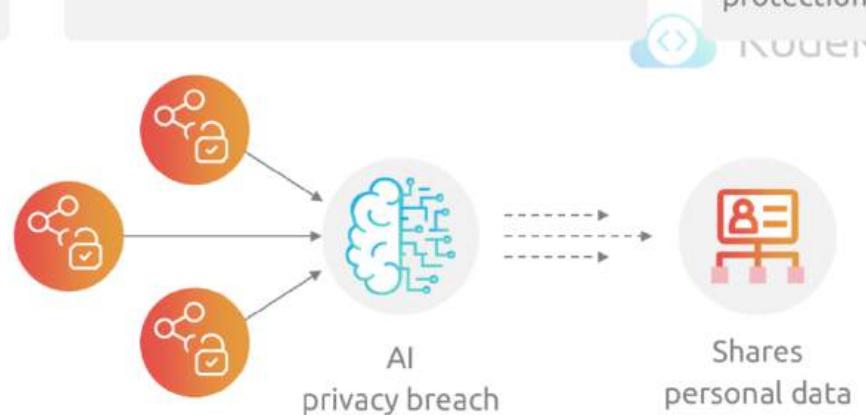
Principles of Responsible AI

Principle	Challenge	Example
Reliability and safety 	Malfunctions or incorrect predictions from AI systems can cause significant harm	An autonomous drone misinterprets its surroundings and crashes, causing property damage and potential injury  

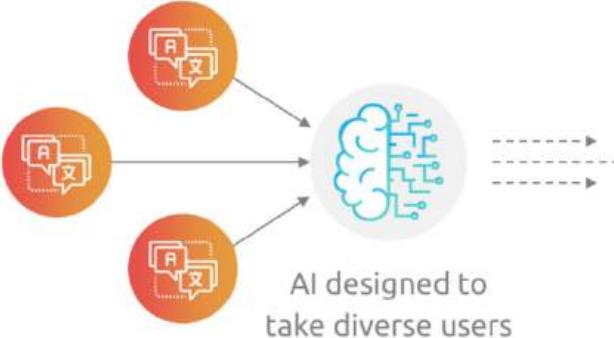
Principles of Responsible AI

Principle	Challenge	Example
Privacy and security 	AI can inadvertently expose or misuse sensitive personal data	A health monitoring app shares private medical information without user consent due to inadequate data protection measures

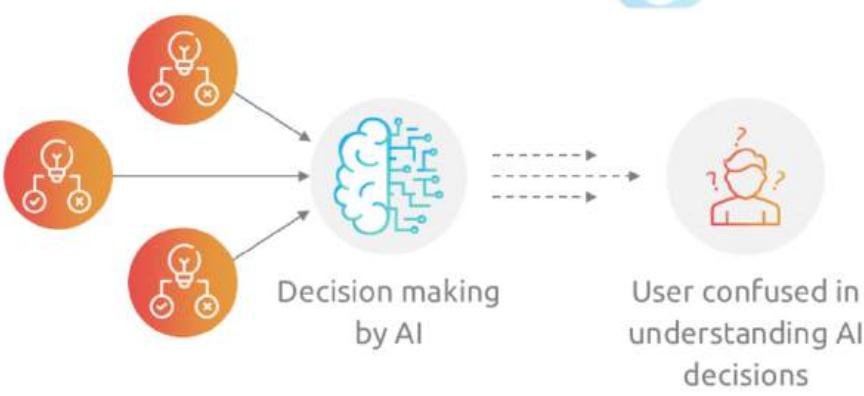


Principles of Responsible AI

Principle	Challenge	Example
Inclusiveness 	AI applications may not be designed to accommodate diverse user needs	A language translation service fails to support dialects commonly spoken in certain communities, excluding those users  

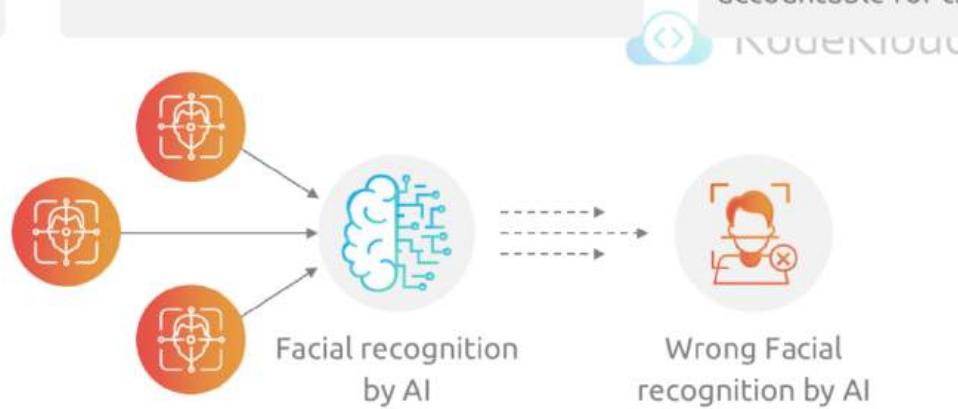

AI designed to take diverse users Fails to support certain languages

Principles of Responsible AI

Principle	Challenge	Example
<p>Transparency</p> 	<p>Users may find it difficult to understand how AI decisions are made</p>  <p>Decision making by AI</p> <p>User confused in understanding AI decisions</p>	<p>An AI-powered credit scoring system provides no explanation for its scores, leaving applicants confused about why they were denied credit</p>  

Principles of Responsible AI

Principle	Challenge	Example
Accountability 	Determining responsibility for AI-driven decisions can be complex	An AI facial recognition system leads to the wrongful arrest of an individual, raising questions about who is accountable for the error



Fundamentals of Machine Learning

Module Introduction

01 What is Machine Learning?

02 Types of Machine Learning

03 Training and Evaluation of Models

04 Deep Learning



KodeKloud

Module Introduction

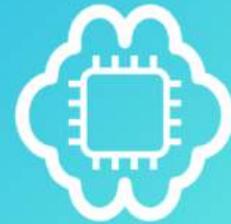
05 Introduction to Azure ML

06 Explore Azure ML

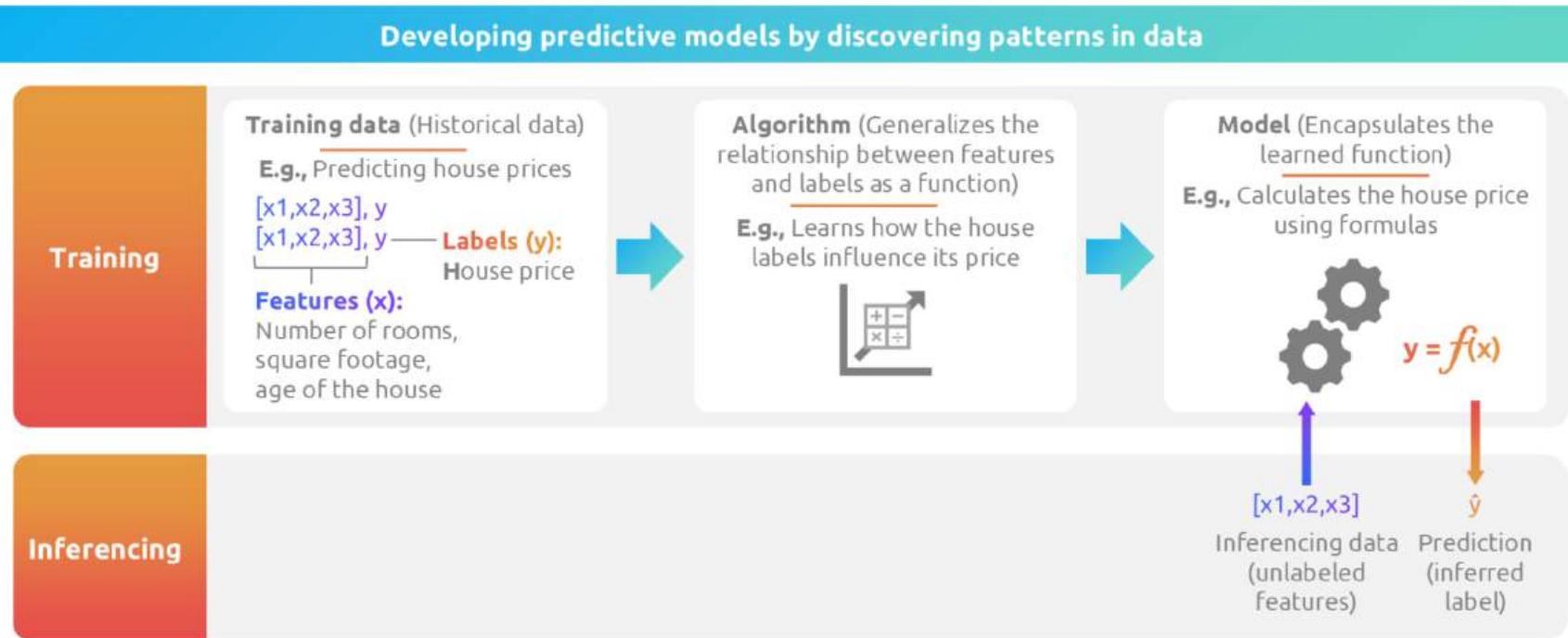


KodeKloud

What is Machine Learning?



What is Machine Learning?



© Copyright KodeKloud

Training Data (Historical observations)

A dataset containing input features (x) and corresponding labels (y). For example, predicting house prices:

Features (x): [number of rooms, square footage, age of the house]

Labels (y): [house price]

Example: [3, 1500, 20], \$300,000

Algorithm (Generalizes the relationship between inputs and labels)

An algorithm processes the training data to find patterns and relationships between the input features (x) and the labels (y).

In our example, the algorithm learns how the number of rooms, square footage, and age of the house influence the house price.

Model (Encapsulates the learned function)

The result of training is a model that encapsulates the function mapping inputs to labels. For instance: $y = f(x)$.

Example function: Price = $(50000 * \text{number of rooms}) + (200 * \text{square footage}) - (1000 * \text{age})$

Inferencing Data (New data without labels)

New data that includes input features (x) but not labels (y). The model uses this data to make predictions.

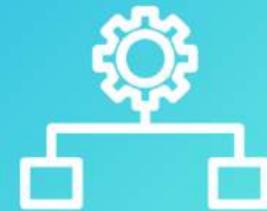
Example: [4, 2000, 10] (4 rooms, 2000 square feet, 10 years old)

Prediction (Inferred label)

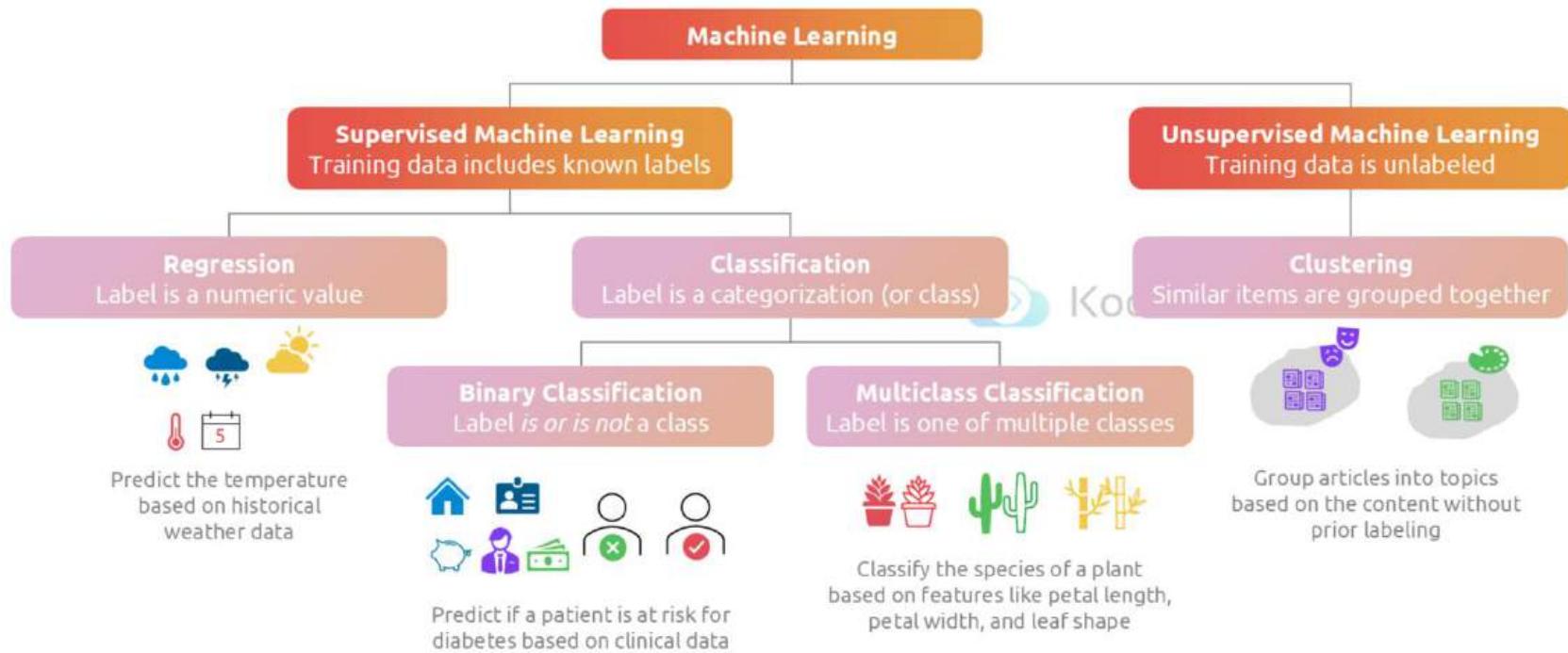
The model generates a predicted label (\hat{y}) for the new data based on the learned function.

Example: The predicted house price is calculated as: Price = $(50000 * 4) + (200 * 2000) - (1000 * 10) = \$400,000$

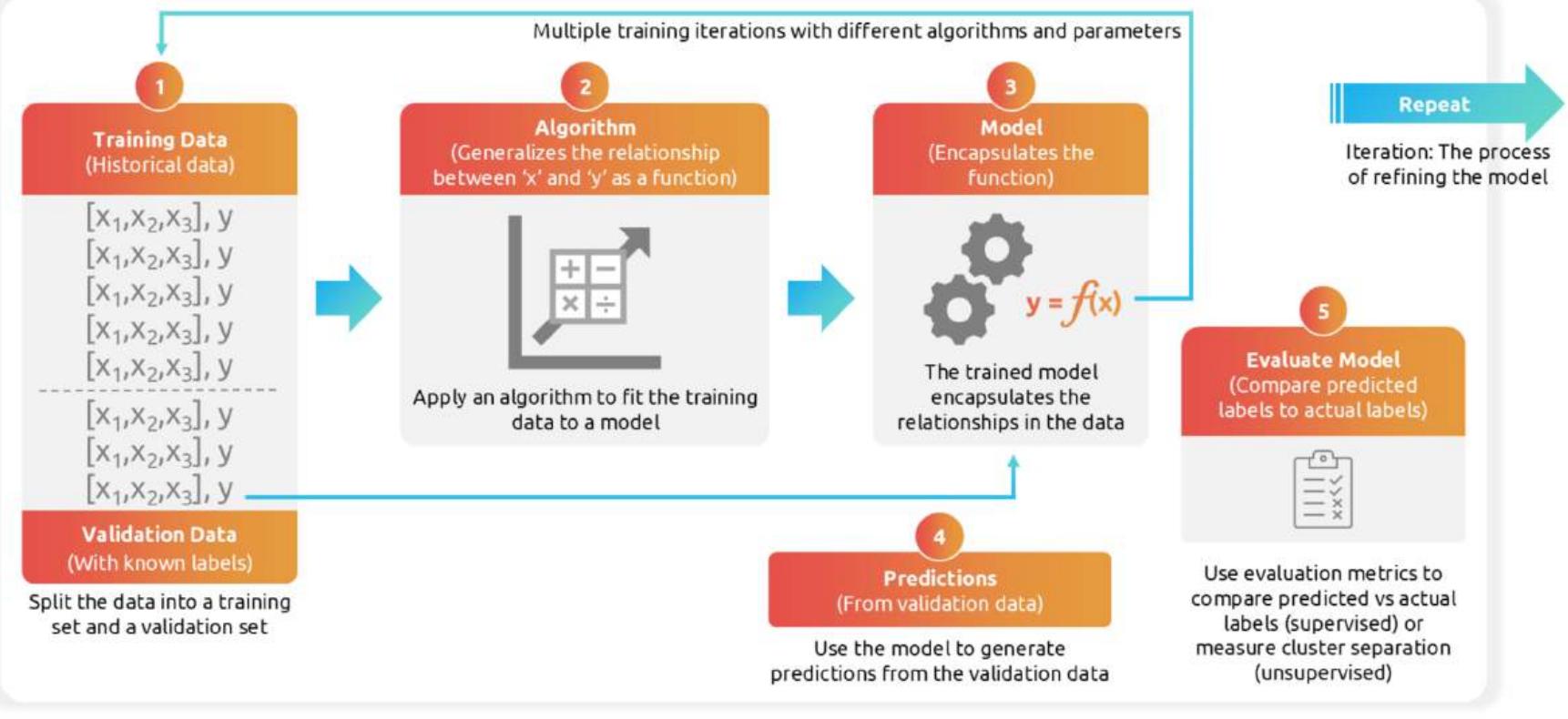
Types of Machine Learning



Types of Machine Learning



Model Training and Evaluation



© Copyright KodeKloud

Split the data into a training set and a validation set

Training Data: Used to train the model. For example, a dataset of house prices including features like size, location, and number of bedrooms.

Validation Data: Used to validate the model's predictions. For example, a separate dataset of house prices with known values for evaluation.

Apply an algorithm to fit the training data to a model

Algorithm: An algorithm processes the training data to learn patterns and relationships. For instance, using linear regression to find the relationship between house features and prices.

The trained model encapsulates the relationships in the data

Model: The output of the training process that represents the learned function. Example: $y = f(x)$, where y is the predicted house price based on features x .

Use the model to generate predictions from the validation data

Predictions: The model uses validation data to predict outcomes. Example: Predicting house prices for new data points in the validation set.

Use evaluation metrics to compare predicted vs actual labels (supervised) or measure cluster separation (unsupervised)

Evaluate Model: Comparing the model's predictions to actual values using metrics like mean absolute error for regression or accuracy for classification. For unsupervised learning, evaluating cluster quality using metrics like silhouette score.

Repeat...

Iteration: The process of refining the model by iterating with different algorithms and parameters to improve accuracy and performance.

Explanation:

The learning component of machine learning occurs during training. We try to capture the relationships between the features and the label in a model. Training is the action of iterating on an algorithm to best fit or encapsulate those relationships.

After training, we have a model that we can test. We use some of the data set aside, validation data, to test how closely our model's predicted labels are to actual labels. In the case of unsupervised learning where we don't know the actual labels, we measure how well our model's predicted labels were at separating clusters. There are many types of evaluation metrics. The important thing to remember is that the goal of machine learning is to find a model that gets as close as possible to predicting the actual label. The best model can still have some margin of error.

Training and Evaluation of Models



Training and Evaluation of Models

Human Neural Network



Neurons respond to electrochemical signals

Neurons in the brain get activated upon receiving electrochemical impulses

Activated neurons transmit signals to others

These impulses travel through synapses to other neurons, creating intricate neural pathways

© Copyright KodeKloud

Human Neural Network

Neurons respond to electrochemical signals

Neurons in the brain activate when they receive electrochemical impulses.

Activated neurons transmit signals to others

These impulses travel through synapses to other neurons, creating intricate neural pathways.

Artificial Neural Network

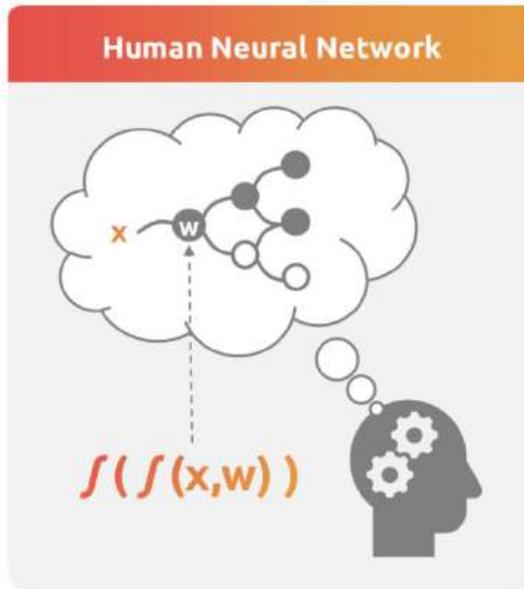
Each artificial neuron processes an input (x) and a weight (w)

In artificial neural networks, each neuron computes an output based on an input value and an associated weight.

Activation functions determine signal propagation

Activation functions in the network decide whether the neuron's output will be passed to subsequent layers.

Training and Evaluation of Models



Human Neural Network

Each artificial neuron processes an input (x) and a weight (w)

In artificial neural networks, each neuron computes an output based on an input value and an associated weight

Activation functions determine signal propagation

Activation functions in the network decide whether the neuron's output will be passed to subsequent layers

© Copyright KodeKloud

Human Neural Network

Neurons respond to electrochemical signals

Neurons in the brain activate when they receive electrochemical impulses.

Activated neurons transmit signals to others

These impulses travel through synapses to other neurons, creating intricate neural pathways.

Artificial Neural Network

Each artificial neuron processes an input (x) and a weight (w)

In artificial neural networks, each neuron computes an output based on an input value and an associated weight.

Activation functions determine signal propagation

Activation functions in the network decide whether the neuron's output will be passed to subsequent layers.

Training and Evaluation of Models

Deep Learning



Advanced form of machine learning that tries to emulate the way the human brain learns



The key to deep learning is the **creation of an artificial neural network** that **simulates electrochemical activity in biological neurons** by using mathematical functions



Artificial neural network

Simulates



Electrochemical activity in neurons

© Copyright KodeKloud

Deep learning is an advanced form of machine learning that tries to emulate the way the human brain learns. The key to deep learning is the creation of an artificial neural network that simulates electrochemical activity in biological neurons by using mathematical functions.

Artificial neural networks are made up of multiple layers of neurons - essentially defining a deeply nested function. This architecture is the reason the technique is referred to as deep learning and the models produced by it are often referred

to as deep neural networks (DNNs). You can use deep neural networks for many kinds of machine learning problem, including regression and classification, as well as more specialized models for natural language processing and computer vision.

Training and Evaluation of Models

Artificial Neural Networks



Made up of multiple layers of neurons often referred to as **Deep Neural Networks (DNNs)**



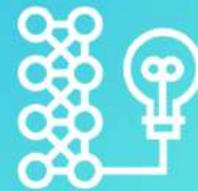
KodeKloud

Used for:

- Machine learning problem including regression and classification
- Natural language processing
- Computer vision

Artificial neural networks are made up of multiple layers of neurons - essentially defining a deeply nested function. This architecture is the reason the technique is referred to as deep learning and the models produced by it are often referred to as deep neural networks (DNNs). You can use deep neural networks for many kinds of machine learning problem, including regression and classification, as well as more specialized models for natural language processing and computer vision.

Deep Learning



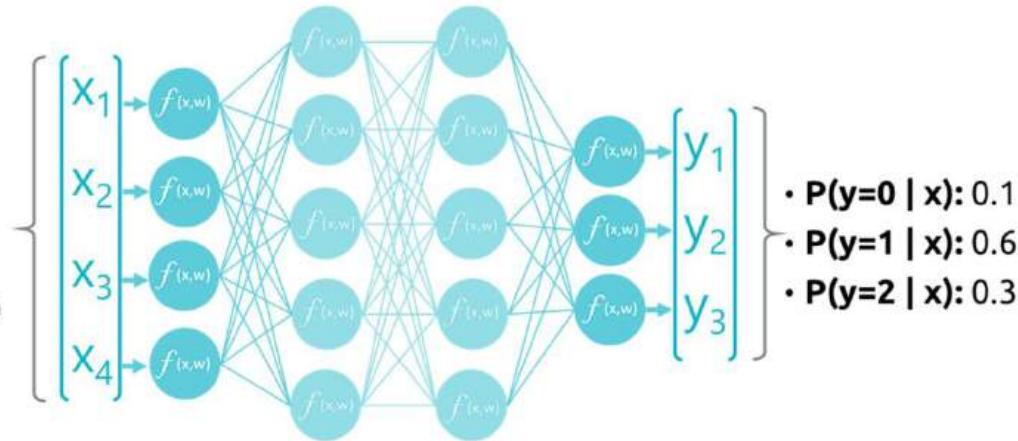
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- x_1 : 220
- x_2 : 0.5
- x_3 : 0.8
- x_4 : 5000



© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

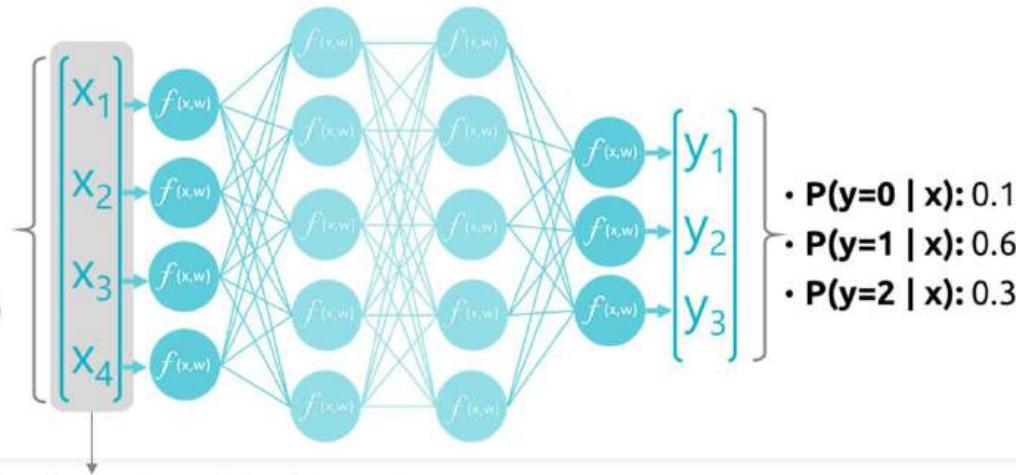
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- x_1 : 220
- x_2 : 0.5
- x_3 : 0.8
- x_4 : 5000



Feature data (x) consists of some characteristics of the sound:

- The pitch of the sound
- The duration of the sound
- The amplitude of the sound
- The frequency of the sound

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

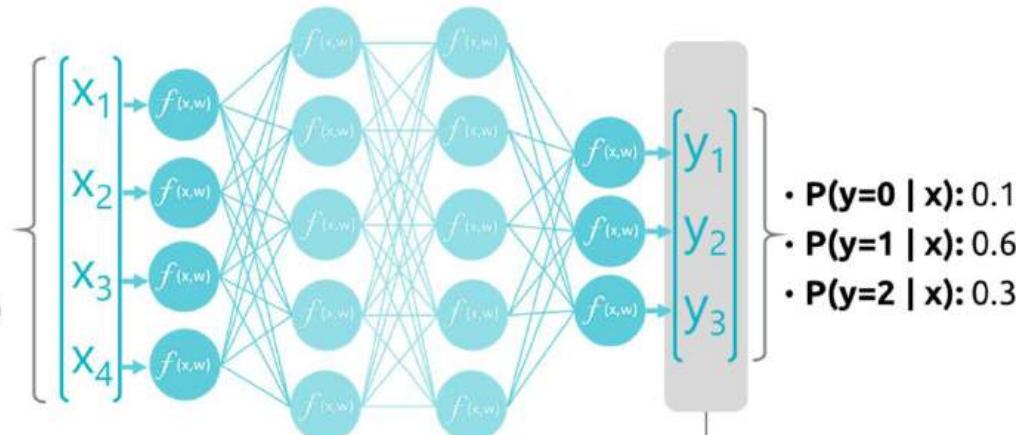
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



Label (y) is type of the animal making the sound:

- Dog
- Cat
- Bird

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

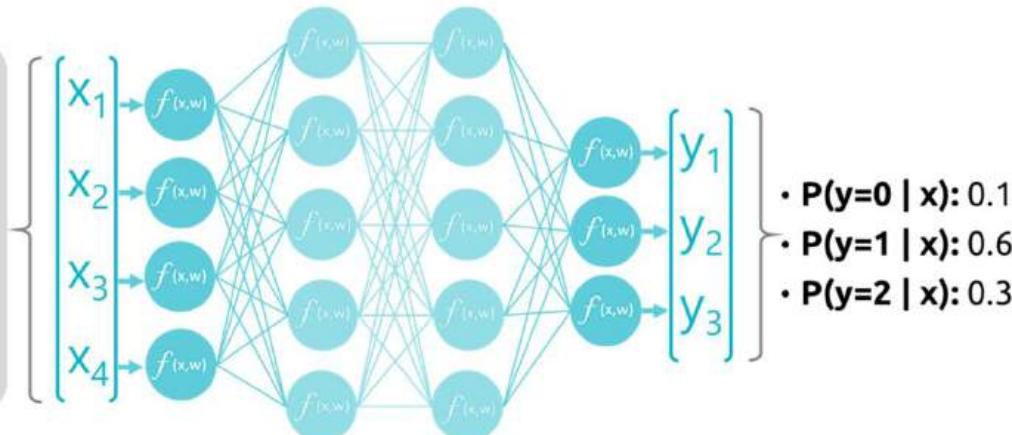
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



The feature vector (x) of a sound observation is fed into the input layer of the neural network, with each x value corresponding to a neuron.



© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

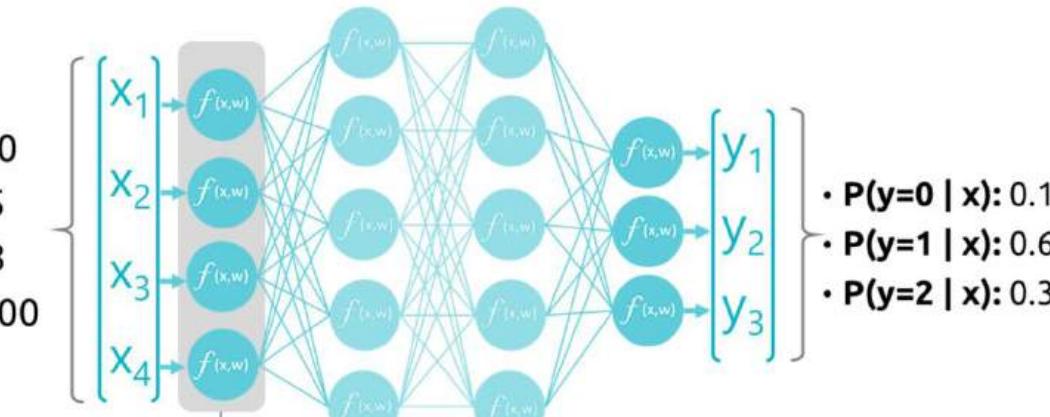
Deep Learning

Deep neural network model

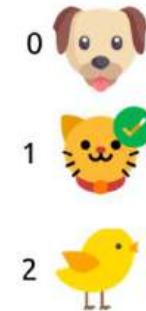
Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



The Functions of the first layer of neurons is to calculate a weighted sum by combining the x value and w weight.



© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

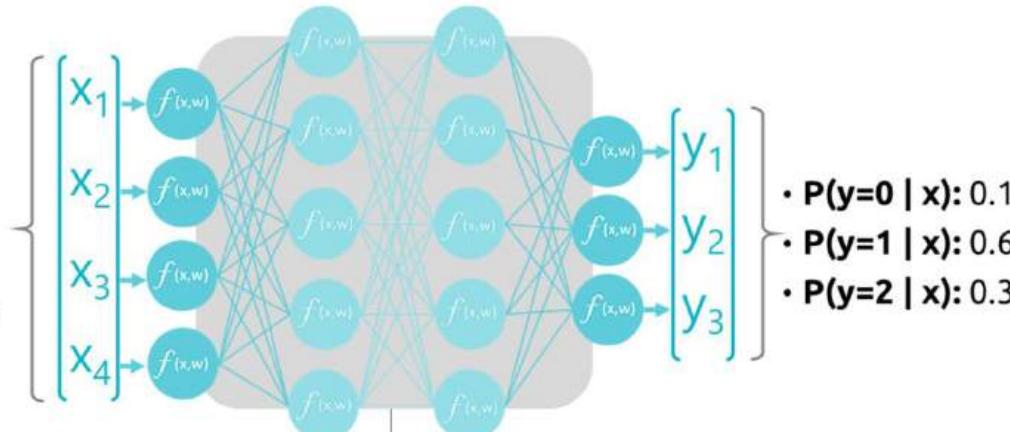
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



Each neuron in a layer is connected to all the neurons in the subsequent layer. The results from each layer are passed forward through the network until they reach the output layer.

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

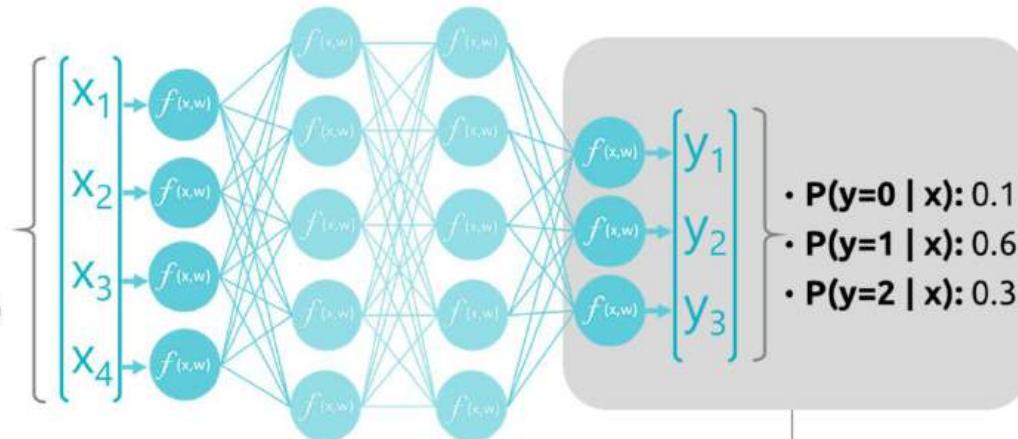
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



The output layer produces a vector of values. Here, it calculates the probability distribution for the three possible classes of animal sounds.

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

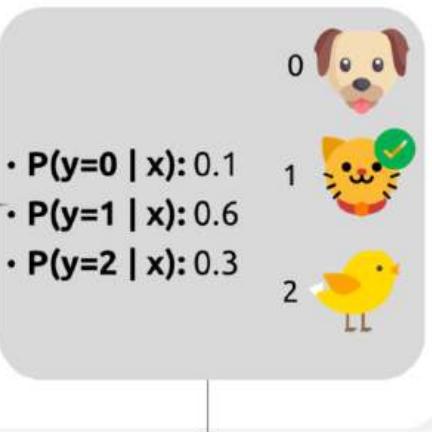
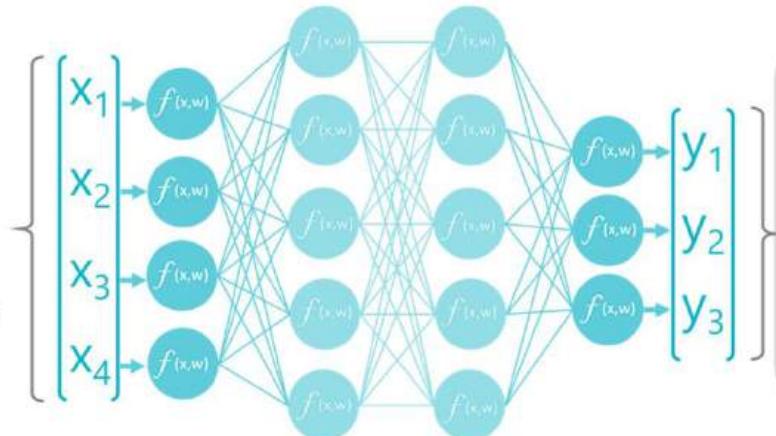
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- x_1 : 220
- x_2 : 0.5
- x_3 : 0.8
- x_4 : 5000



The elements of the vector represent the probabilities for classes 0, 1, and 2.

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

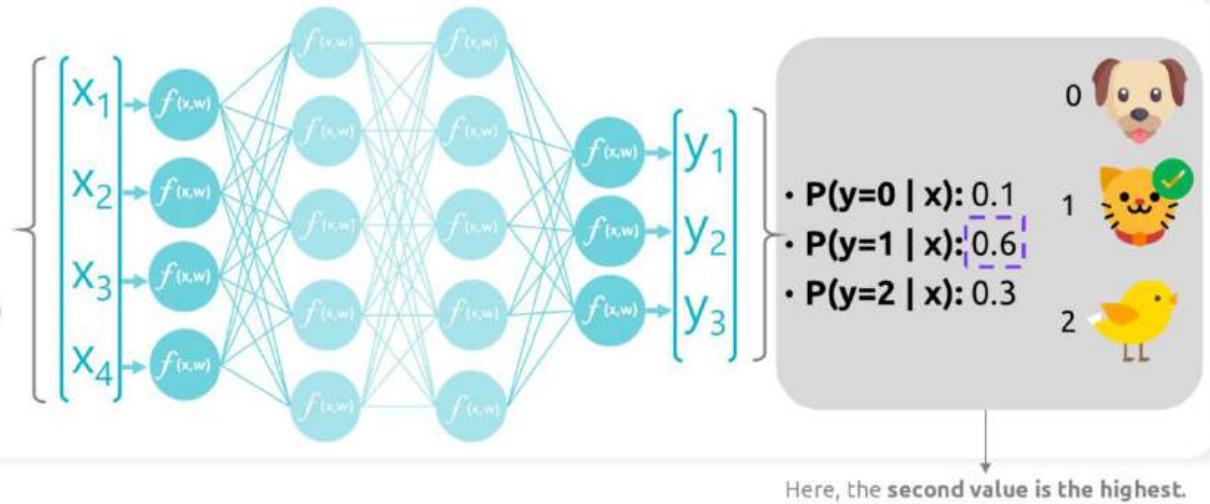
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- x_1 : 220
- x_2 : 0.5
- x_3 : 0.8
- x_4 : 5000



© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

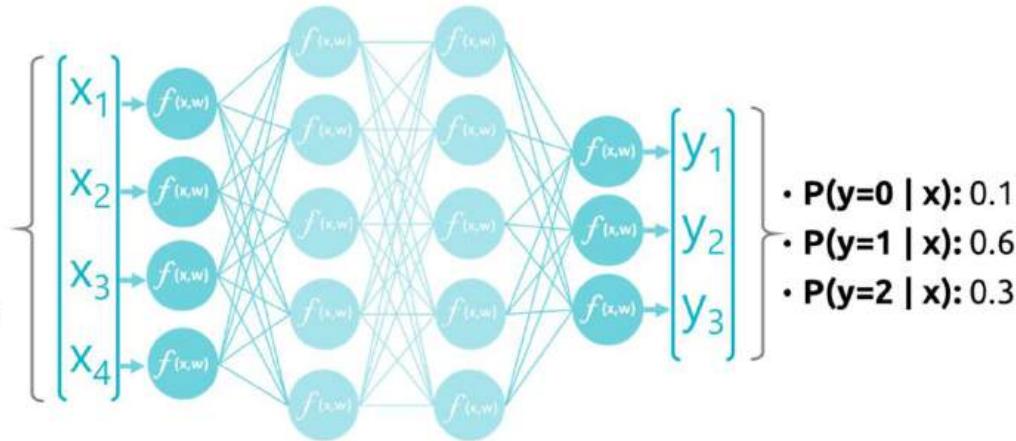
Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.



- $x_1: 220$
- $x_2: 0.5$
- $x_3: 0.8$
- $x_4: 5000$



The model predicts that the type of animal sound is 1 (Cat).

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

Deep Learning

Deep neural network model

Let's explore an example in which a neural network is used to define a classification model for animal sounds.

CONCLUSION

- The model learns the correct weights by repeatedly passing the training data through the network and comparing the predicted probabilities for each class with the actual class label.
- The weights are adjusted in each iteration to reduce the error (or loss).
- The weights are adjusted between the predicted probabilities (like [0.1, 0.6, 0.3]) and the actual class label (like [0.0, 1.0, 0.0] for class 1, which could represent a Cat).

© Copyright KodeKloud

To better understand how a deep neural network model works, let's explore an example in which a neural network is used to define a classification model for animal sounds.

The feature data (x) consists of some characteristics of the sound. Specifically, the measurements are:

- The pitch of the sound.
- The duration of the sound.

- The amplitude of the sound.
- The frequency of the sound.

The label we're trying to predict (y) is the type of animal making the sound, and there are three possible types it could be:

- Dog
- Cat
- Bird

The process for inferencing a predicted animal sound class using this network is:

The feature vector for a sound observation is fed into the input layer of the neural network, which consists of a neuron for each x value. In this example, the following x vector is used as the input: [220, 0.5, 0.8, 5000].

The functions for the first layer of neurons each calculate a weighted sum by combining the x value and w weight.

Each neuron in a layer is connected to all of the neurons in the next layer. The results of each layer are fed forward through the network until they reach the output layer.

The output layer produces a vector of values; in this case, using a softmax or similar function to calculate the probability distribution for the three possible classes of animal sounds. In this example, the output vector is: [0.1, 0.6, 0.3].

The elements of the vector represent the probabilities for classes 0, 1, and 2. The second value is the highest, so the model predicts that the type of animal sound is 1 (Cat).

The model learns the appropriate weights by iteratively passing the training data through and evaluating the predicted probabilities for each class against the known class label. The weights are adjusted in each iteration to reduce the amount of error (loss) between the predicted probabilities (for example [0.1, 0.6, 0.3]) and the values that represent the actual class label (for example [0.0, 1.0, 0.0] for class 1 (Cat)).

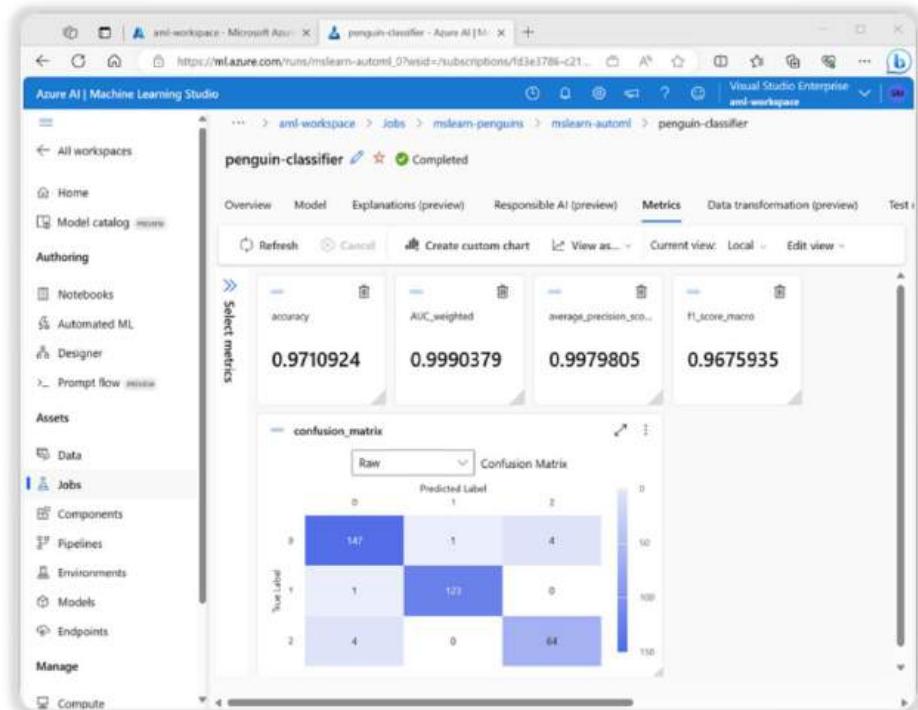
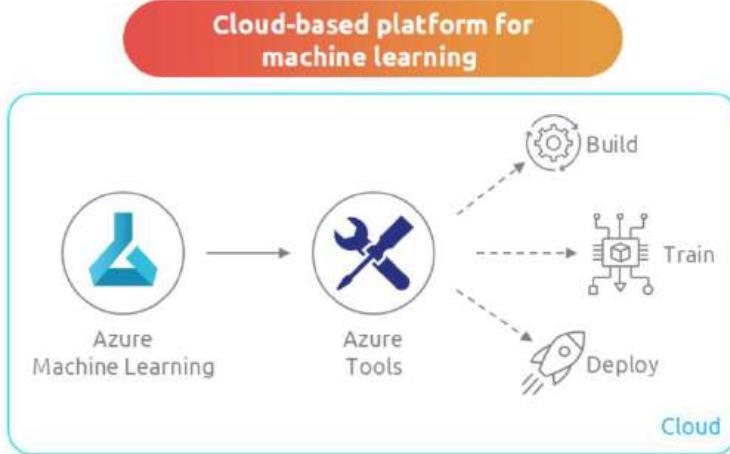
Azure ML – Introduction



Azure Machine Learning – Introduction

The screenshot shows the Azure Machine Learning Studio interface. The left sidebar navigation includes 'All workspaces', 'Home', 'Model catalog', 'Authoring' (with 'Notebooks', 'Automated ML', 'Designer', and 'Prompt flow' sub-options), 'Assets' (with 'Data', 'Jobs', 'Components', 'Pipelines', 'Environments', 'Models', and 'Endpoints'), and 'Manage' (with 'Compute'). The main content area displays the 'penguin-classifier' model, which is 'Completed'. The 'Metrics' tab is selected, showing four performance metrics: accuracy (0.9710924), AUC_weighted (0.9990379), average_precision_score (0.9979805), and f1_score_macro (0.9675935). Below these metrics is a 'confusion_matrix' visualization. The confusion matrix table has 'Predicted Label' columns (0, 1, 2) and 'True Label' rows (0, 1, 2). The values in the matrix are: (0,0) 547, (0,1) 1, (0,2) 4; (1,0) 1, (1,1) 123, (1,2) 0; (2,0) 4, (2,1) 0, (2,2) 64. A color scale bar on the right indicates values from 0 to 150.

Azure Machine Learning – Introduction

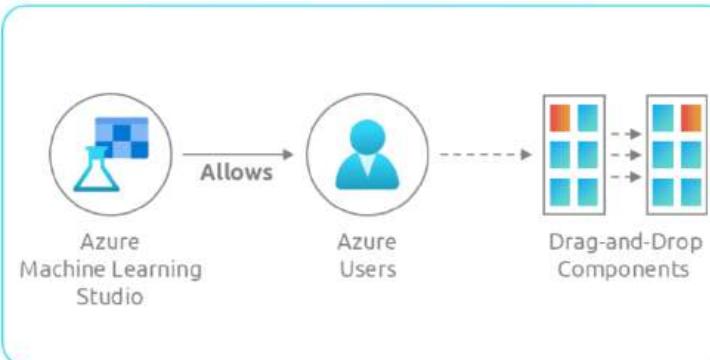


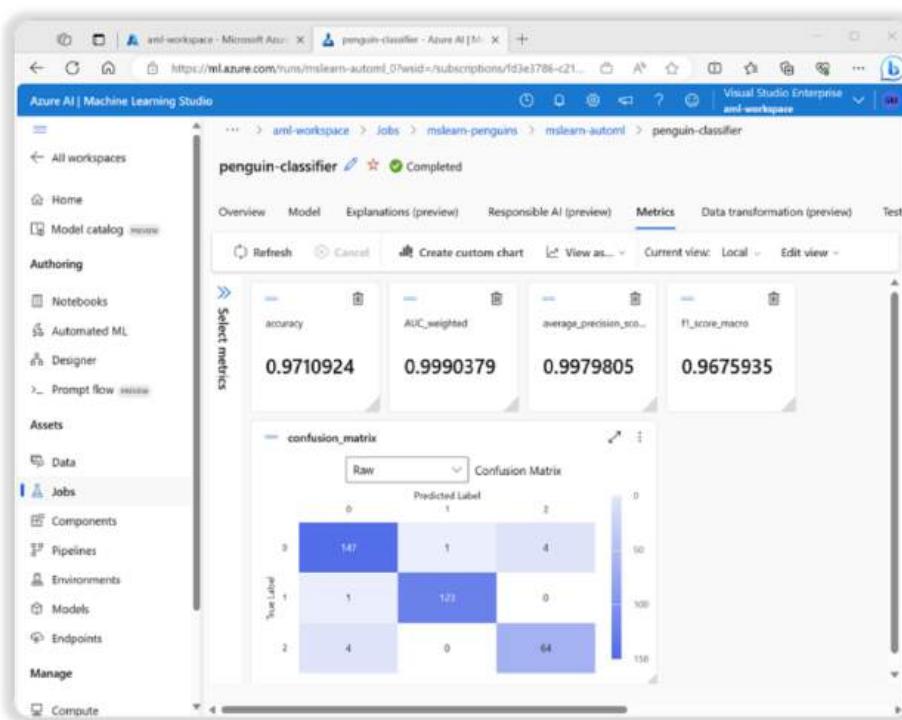
© Copyright KodeKloud

Cloud-Based Platform for Machine Learning : Azure Machine Learning provides a comprehensive platform hosted in the cloud, offering tools and services to build, train, and deploy machine learning models.

Azure Machine Learning – Introduction

User-friendly interface:
Azure Machine Learning Studio





The screenshot shows the Azure AI | Machine Learning Studio interface. The top navigation bar includes 'Overview', 'Model', 'Explanations (preview)', 'Responsible AI (preview)', 'Metrics' (which is selected), 'Data transformation (preview)', and 'Test'. Below the navigation is a table of metrics:

metric	value
accuracy	0.9710924
AUC_weighted	0.9990379
average_precision_score	0.9979805
f1_score_macro	0.9675935

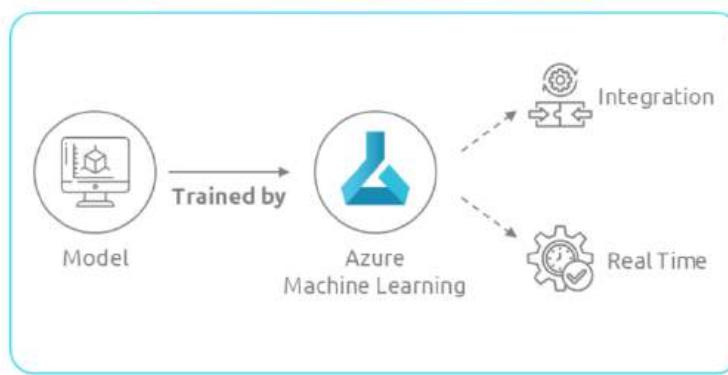
Below the metrics is a 'confusion_matrix' section with a heatmap. The columns are labeled 'Predicted Label' (0, 1, 2) and the rows are labeled 'True Label' (0, 1, 2). The values in the matrix are:

		Predicted Label		
		0	1	2
True Label	0	147	1	4
	1	1	123	0
	2	4	0	64

© Copyright KodeKloud

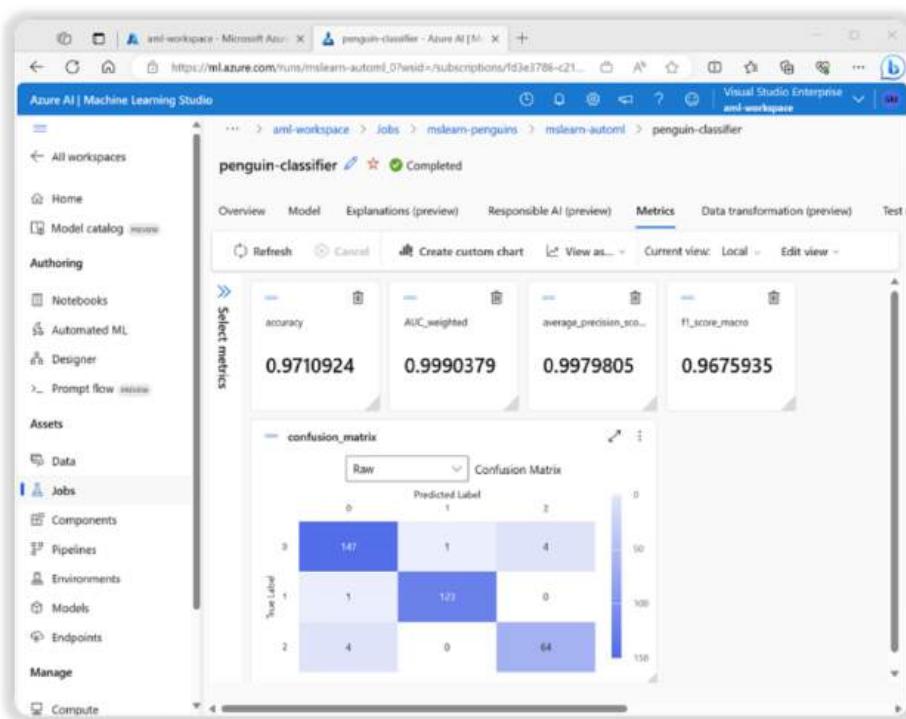
User-Friendly Interface: Azure Machine Learning Studio: Azure Machine Learning Studio is a graphical interface that simplifies access to machine learning capabilities, allowing users to drag-and-drop components to create models without writing extensive code.

Azure Machine Learning – Introduction



Deploy machine learning models as services

The diagram illustrates the deployment process. A 'Model' (represented by a computer monitor icon) is 'Trained by' Azure Machine Learning (represented by a blue gear icon). This trained model then 'Integrates' with 'Real Time' predictions (represented by a gear with a checkmark icon).



Screenshot of the Azure AI | Machine Learning Studio Metrics page for a penguin-classifier model. The page shows various performance metrics and a confusion matrix.

Metric	Value
accuracy	0.9710924
AUC_weighted	0.9990379
average_precision_score	0.9979805
f1_score_macro	0.9675935

Confusion Matrix:

		Predicted Label		
		0	1	2
True Label	0	147	1	4
	1	1	123	0
2	4	0	64	

© Copyright KodeKloud

Deploy Machine Learning Models as Services: Models trained using Azure Machine Learning can be easily deployed as web services, making them accessible for integration into applications and enabling real-time predictions

Azure AI Services – Introduction



Module Introduction

01 Fundamentals of Azure AI

02 Azure AI Services

03 Explore Azure AI Services



KodeKloud

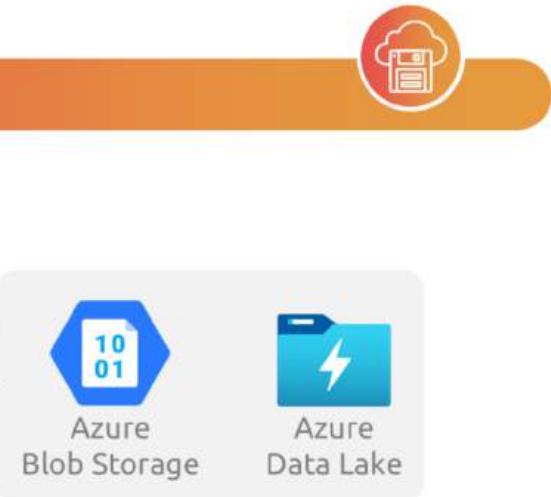
Fundamentals of Azure AI



Fundamentals of Azure AI



Data Storage

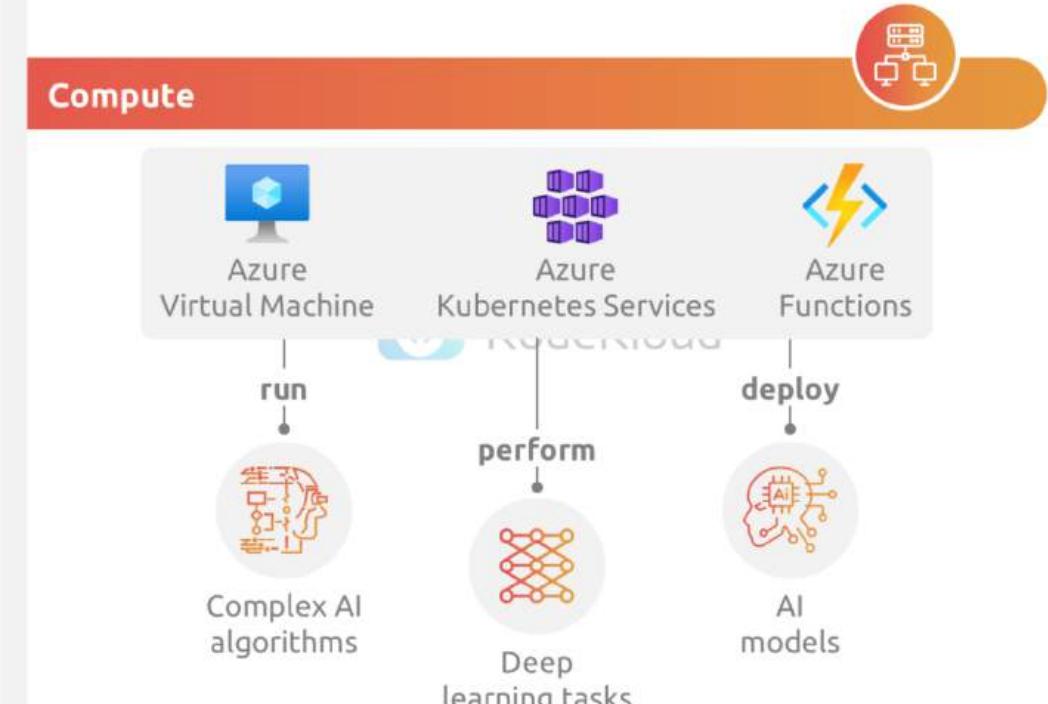


It ensures that data is **securely stored** and **easily accessible** for training and running AI models.

© Copyright KodeKloud

Data Storage: Azure's data storage solutions, such as Azure Blob Storage and Azure Data Lake, are vital for storing the vast amounts of data required for AI applications. These services ensure that data is securely stored and easily accessible for training and running AI models.

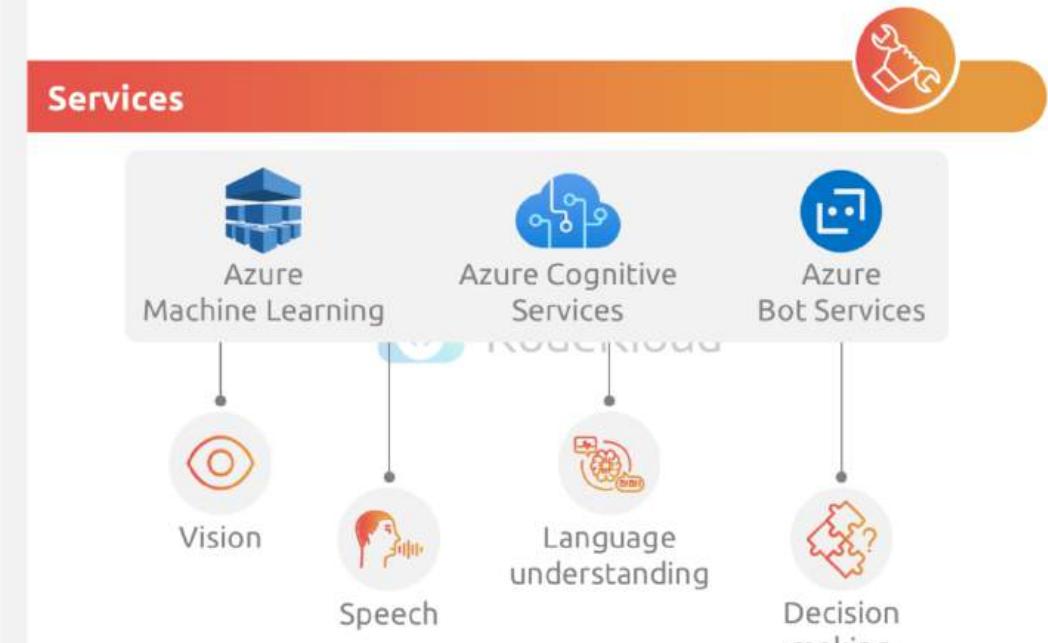
Fundamentals of Azure AI



© Copyright KodeKloud

Compute: Azure offers powerful compute resources like Azure Virtual Machines, Azure Kubernetes Service (AKS), and Azure Functions. These resources provide the necessary processing power to run complex AI algorithms, perform deep learning tasks, and deploy AI models at scale

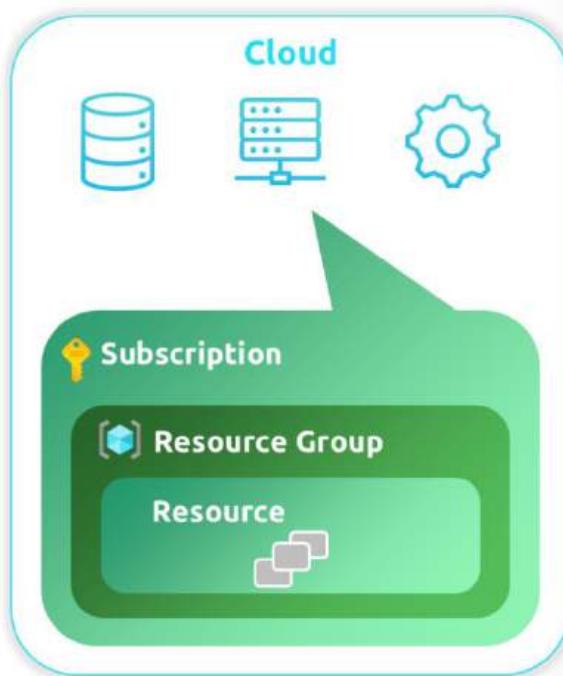
Fundamentals of Azure AI



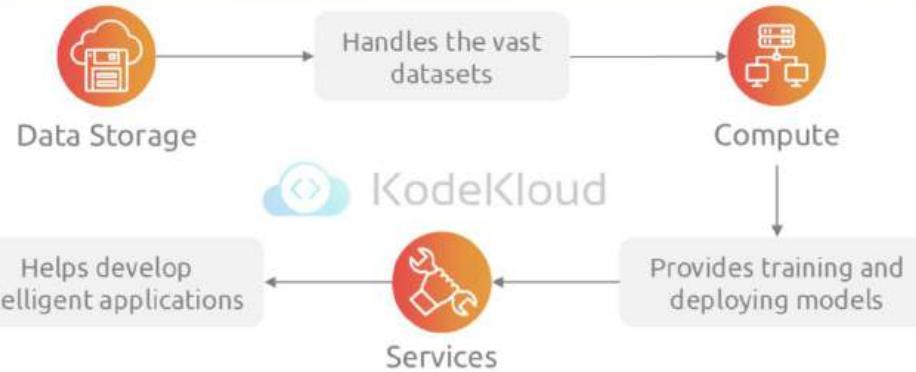
© Copyright KodeKloud

Services: Azure encompasses a variety of AI services, including Azure Machine Learning, Azure Cognitive Services, and Azure Bot Services. These services provide ready-made AI capabilities such as vision, speech, language understanding, decision-making, and more, enabling developers to integrate sophisticated AI functionalities into their applications easily.

Fundamentals of Azure AI



Conclusion



Together, they create a comprehensive ecosystem for building and scaling AI solutions on Azure.

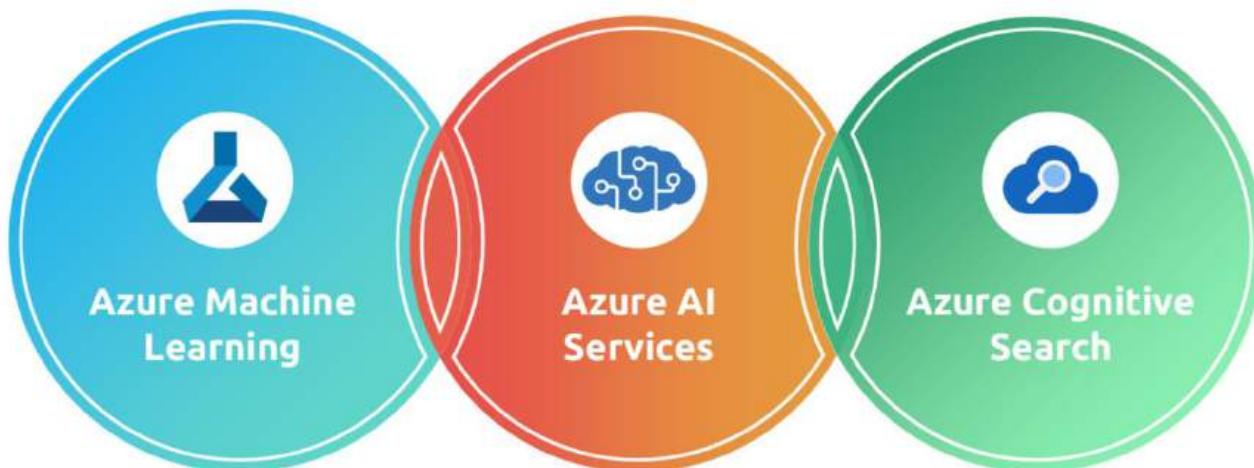
© Copyright KodeKloud

Azure's foundational components—data storage, compute, and services—play a crucial role in enabling Azure AI services. Data storage handles the vast datasets essential for AI, compute resources provide the power needed for training and deploying models, and specialized AI services offer pre-built capabilities that accelerate the development of intelligent applications. Together, these elements create a comprehensive ecosystem for building and scaling AI solutions on Azure.

Azure AI Services



AI Services in Azure



AI Services in Azure

Azure Machine Learning



- Comprehensive platform designed for the **training, deployment, and management** of machine learning models
- Supports a wide range of **machine learning algorithms**
- Facilitates **easy integration** into applications

A comprehensive platform designed for the training, deployment, and management of machine learning models. It supports a wide range of machine learning algorithms and facilitates easy integration into applications.

AI Services in Azure

Azure AI Services



- **Collection of services** with capabilities for Vision, Speech, Language, Decision-making, and Generative AI
- Provides powerful tools for **building intelligent applications**
- **Enables applications** to see, hear, understand, and generate content

A collection of services that include capabilities for Vision, Speech, Language, Decision-making, and Generative AI. These services provide powerful tools for building intelligent applications that can see, hear, understand, and even generate content.

AI Services in Azure

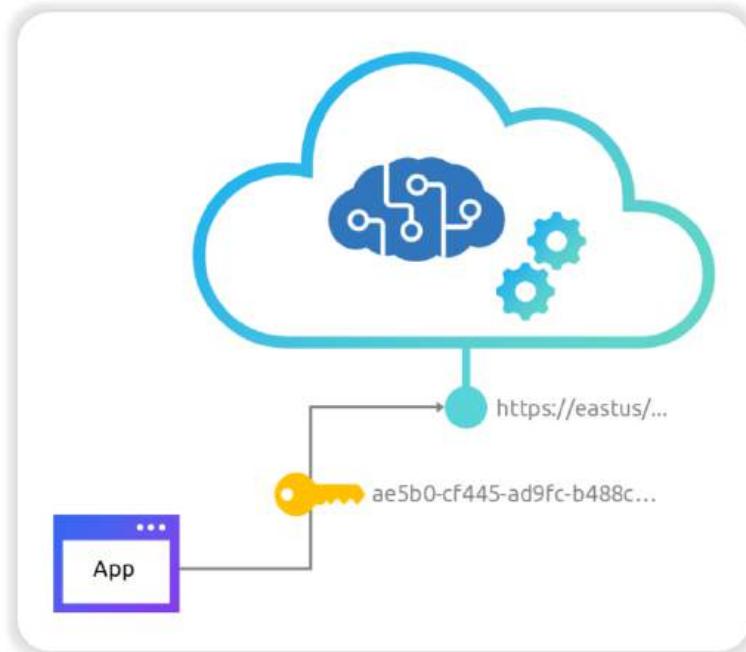
Azure Cognitive Search



- Advanced service for **data extraction, enrichment, and indexing**
- **Enhances the search experience** with intelligent search capabilities
- **Facilitates knowledge** discovery through effective data mining techniques

An advanced service for data extraction, enrichment, and indexing. It enhances the search experience by providing intelligent search capabilities and facilitating knowledge discovery through effective data mining techniques.

AI Services in Azure



KodeKloud

© Copyright KodeKloud

AI Application Resources in an Azure Subscription

Standalone Resources for Specific Services: Dedicated resources that provide specific AI capabilities, such as Azure Cognitive Services for vision, speech, language, and decision-making.

General Azure AI Services Resource for Multiple Services: A unified resource that supports various AI services, allowing for more flexible and comprehensive AI solutions.

Consumed by Applications via:

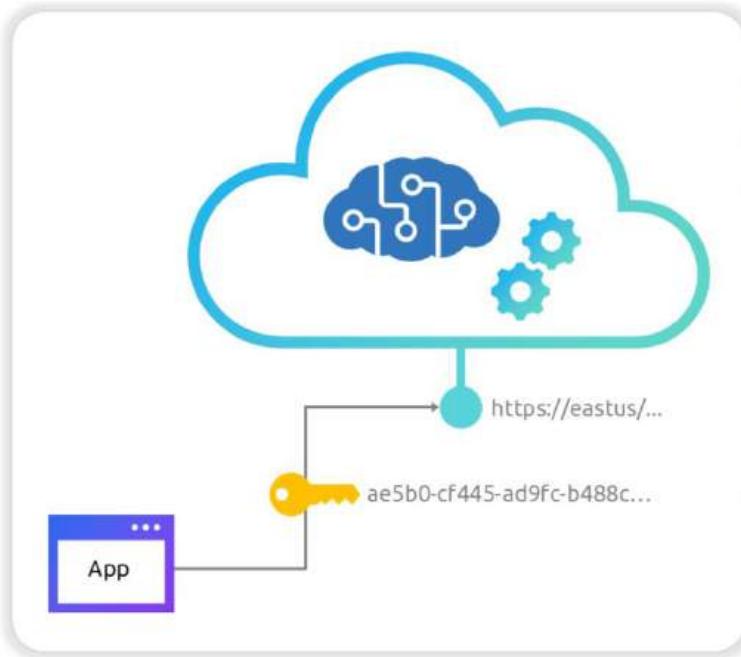
A REST Endpoint (<https://address>): AI services are accessed through RESTful APIs, which provide a standardized way for applications to interact with AI capabilities over the internet.

An Authentication Key or Authorization Token: Secure access to AI services is ensured through the use of authentication keys or tokens, which verify the identity of the requesting application and allow it to consume the services.

Explanation:

Azure AI Services provide a scalable and flexible platform for integrating AI capabilities into applications. By utilizing REST endpoints, developers can easily connect their applications to powerful AI services hosted on Azure. Authentication keys and tokens ensure secure and controlled access, making it possible to deploy intelligent features such as image recognition, natural language processing, and predictive analytics. Whether using standalone resources for specific tasks or a general resource for multiple services, Azure AI Services streamline the integration of advanced AI functionalities into diverse applications.

AI Services in Azure



AI Application Resources in an Azure Subscription

Standalone Resources for specific services

- Provide specific AI capabilities such as Azure Cognitive Services for
 - vision
 - speech
 - language
 - decision-making



General Azure AI Services Resource for multiple services

- A unified resource that supports various AI services
- Allows more flexible and comprehensive AI solutions

© Copyright KodeKloud

AI Application Resources in an Azure Subscription

Standalone Resources for Specific Services: Dedicated resources that provide specific AI capabilities, such as Azure Cognitive Services for vision, speech, language, and decision-making.

General Azure AI Services Resource for Multiple Services: A unified resource that supports various AI services, allowing for more flexible and comprehensive AI solutions.

Consumed by Applications via:

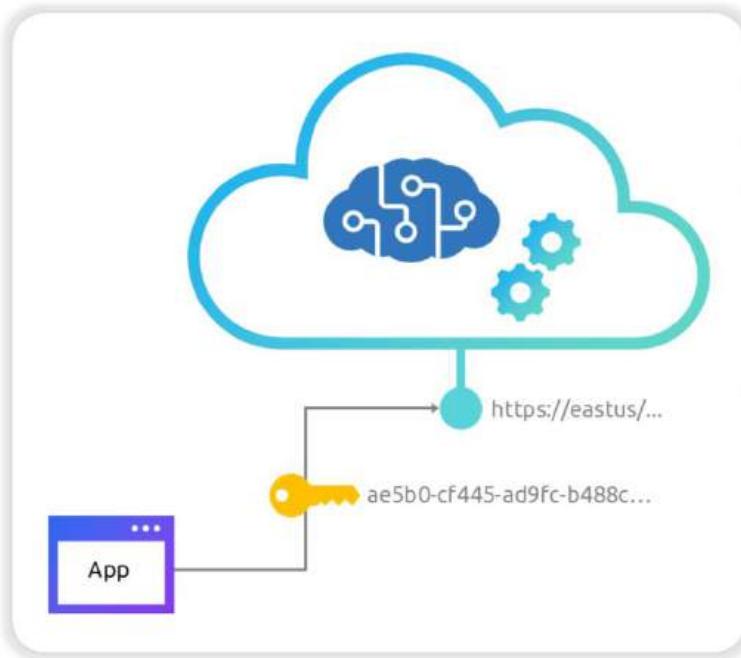
A REST Endpoint (<https://address>): AI services are accessed through RESTful APIs, which provide a standardized way for applications to interact with AI capabilities over the internet.

An Authentication Key or Authorization Token: Secure access to AI services is ensured through the use of authentication keys or tokens, which verify the identity of the requesting application and allow it to consume the services.

Explanation:

Azure AI Services provide a scalable and flexible platform for integrating AI capabilities into applications. By utilizing REST endpoints, developers can easily connect their applications to powerful AI services hosted on Azure. Authentication keys and tokens ensure secure and controlled access, making it possible to deploy intelligent features such as image recognition, natural language processing, and predictive analytics. Whether using standalone resources for specific tasks or a general resource for multiple services, Azure AI Services streamline the integration of advanced AI functionalities into diverse applications.

AI Services in Azure



Consumed by Applications

A REST Endpoint (<https://service-address>)

- AI services are accessed through **RESTful APIs**.
- It provides a **standardized way** for applications to interact with AI capabilities over the internet.

An Authentication Key or Authorization Token

- Secure access to AI services is ensured using **authentication keys or tokens**.
- It verifies the identity of the requesting application and allows it to consume the services.

© Copyright KodeKloud

AI Application Resources in an Azure Subscription

Standalone Resources for Specific Services: Dedicated resources that provide specific AI capabilities, such as Azure Cognitive Services for vision, speech, language, and decision-making.

General Azure AI Services Resource for Multiple Services: A unified resource that supports various AI services, allowing for more flexible and comprehensive AI solutions.

Consumed by Applications via:

A REST Endpoint (<https://address>): AI services are accessed through RESTful APIs, which provide a standardized way for applications to interact with AI capabilities over the internet.

An Authentication Key or Authorization Token: Secure access to AI services is ensured through the use of authentication keys or tokens, which verify the identity of the requesting application and allow it to consume the services.

Explanation:

Azure AI Services provide a scalable and flexible platform for integrating AI capabilities into applications. By utilizing REST endpoints, developers can easily connect their applications to powerful AI services hosted on Azure. Authentication keys and tokens ensure secure and controlled access, making it possible to deploy intelligent features such as image recognition, natural language processing, and predictive analytics. Whether using standalone resources for specific tasks or a general resource for multiple services, Azure AI Services streamline the integration of advanced AI functionalities into diverse applications.

Concepts of Computer Vision

Module Introduction

- 01 Image Processing
- 02 Convolutional Neural Networks
- 03 Multi-Modal Models
- 04 Azure Computer Vision Services



KodeKloud

Image Processing



Artificial Intelligence

Grayscale Image

Consists of an array of pixel values between 0 and 255



0



255

Color Image

A color picture consists of three arrays called **channels**

Each contains pixel values for



A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Artificial Intelligence

An image is an array of pixel values

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	255	255	255	0	0
0	0	255	255	255	0	0
0	0	255	255	255	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Filters are applied to change images

-1	-1	-1	0	0	0	0
-1	8	-1	0	0	0	0
-1	-1	255	255	255	0	0
0	0	255	0	255	0	0
0	0	255	255	255	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

© Copyright KodeKloud

A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

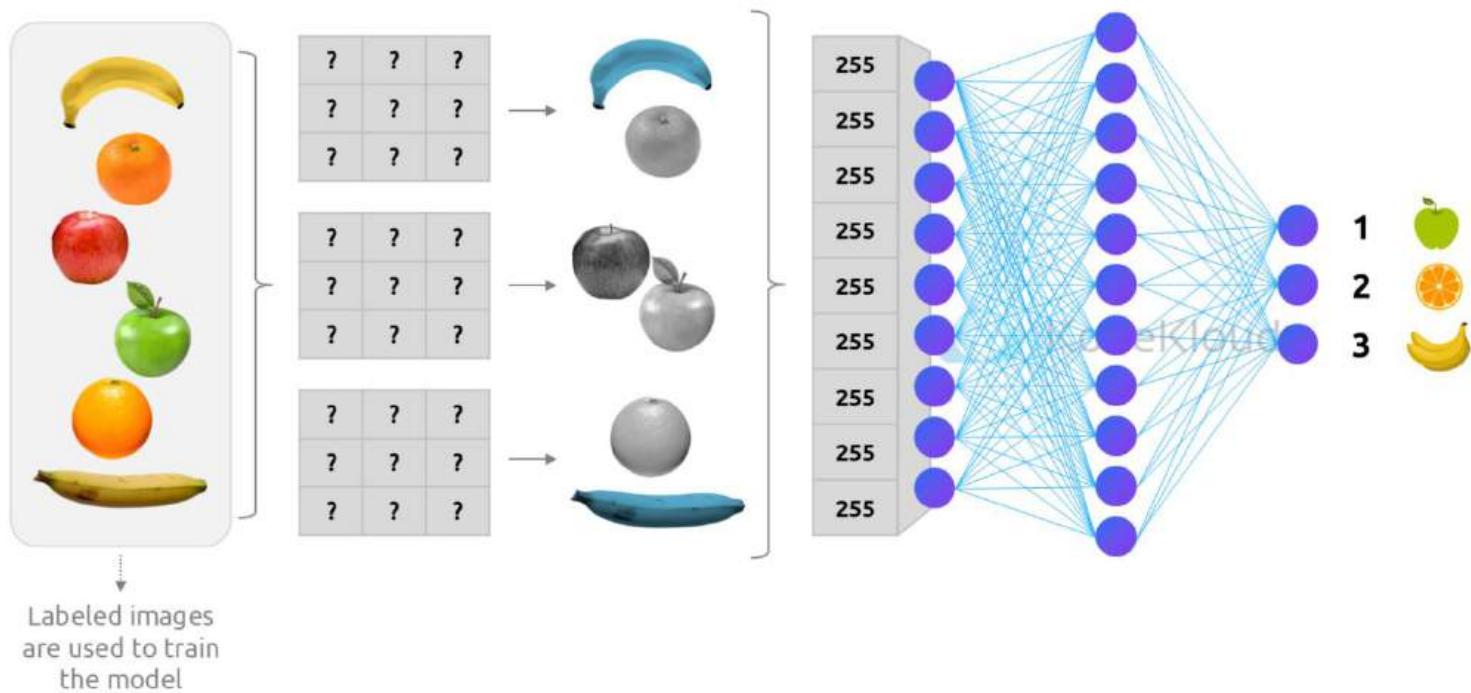
The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



Convolutional Neural Networks



© Copyright KodeKloud

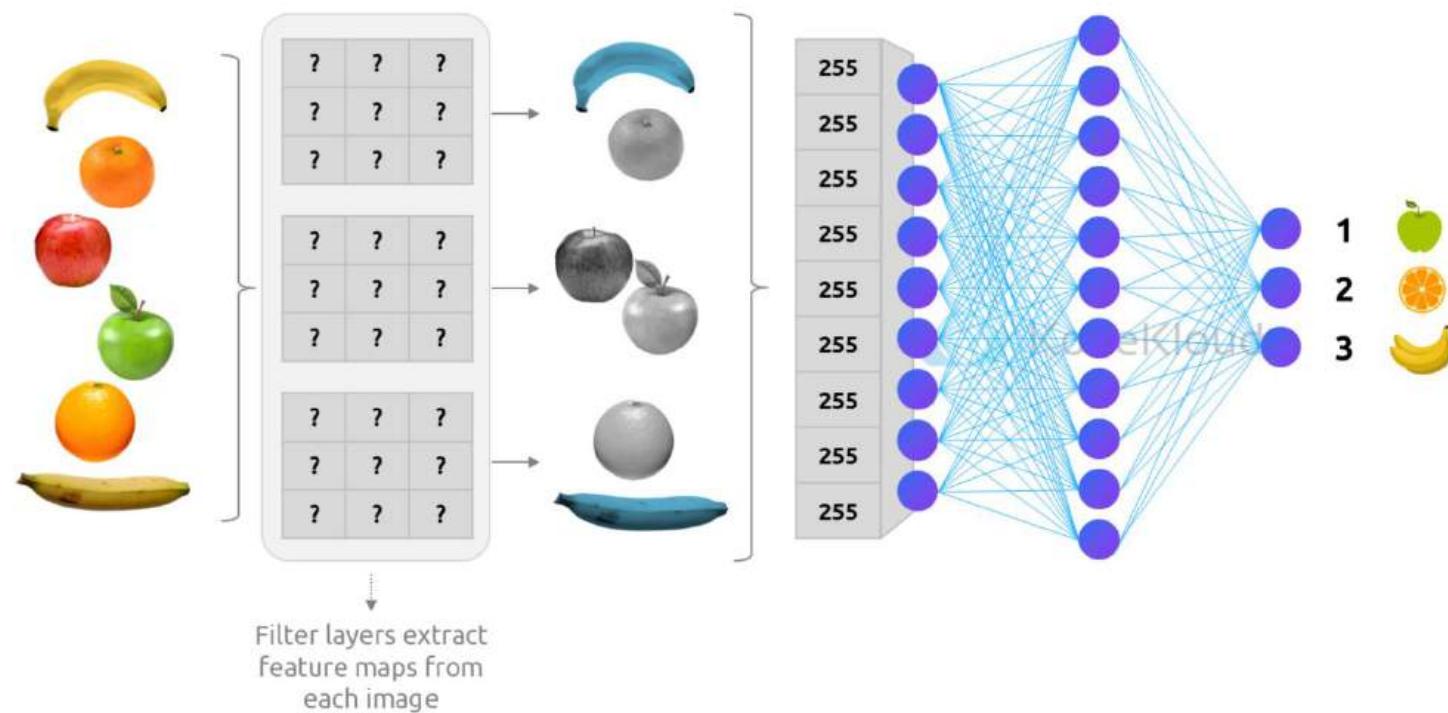
A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



© Copyright KodeKloud

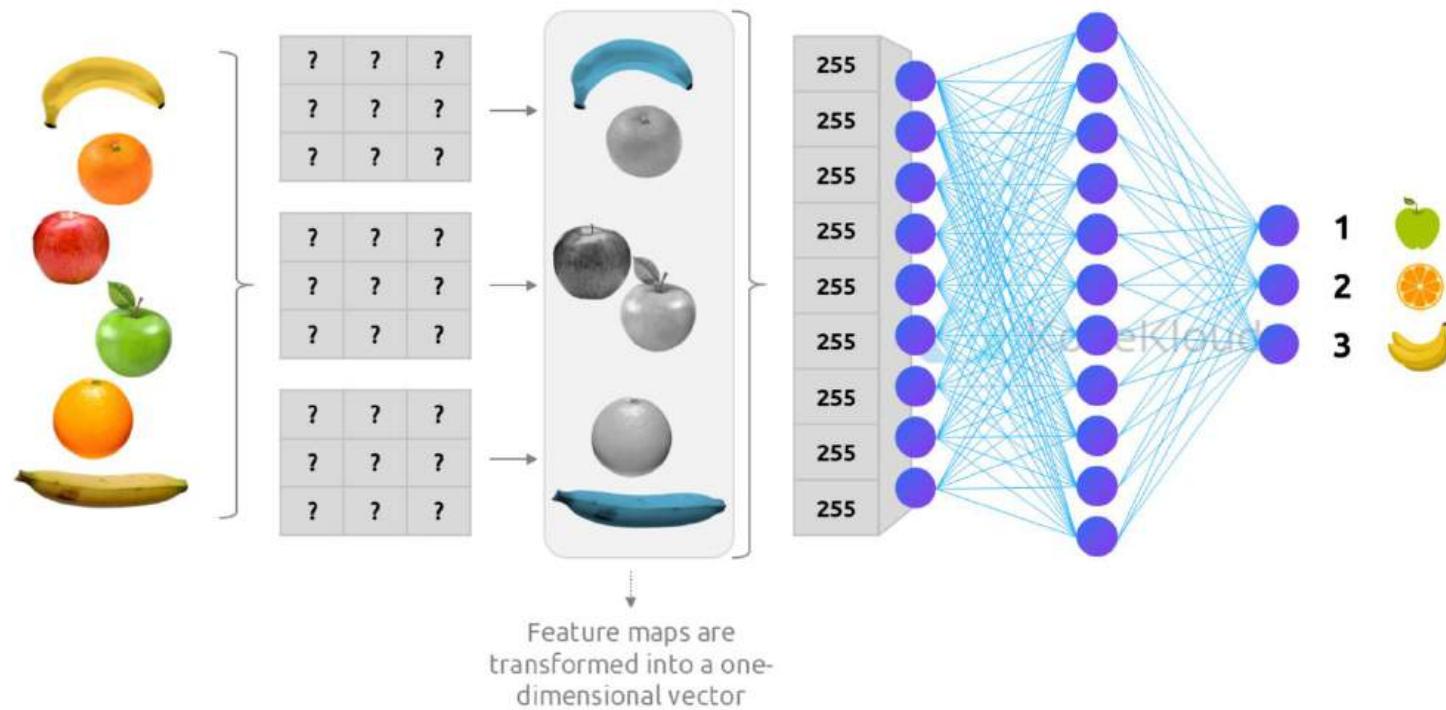
A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



© Copyright KodeKloud

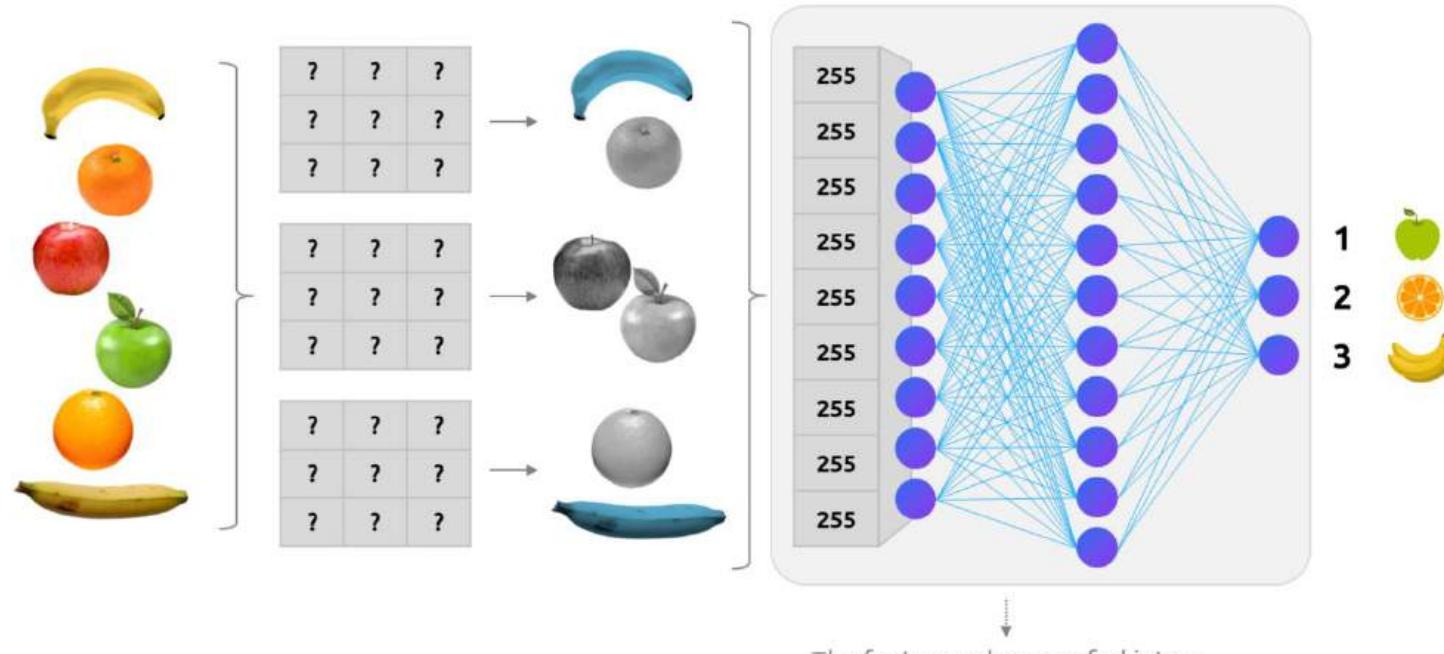
A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



The feature values are fed into a
Fully connected Neural Network

© Copyright KodeKloud

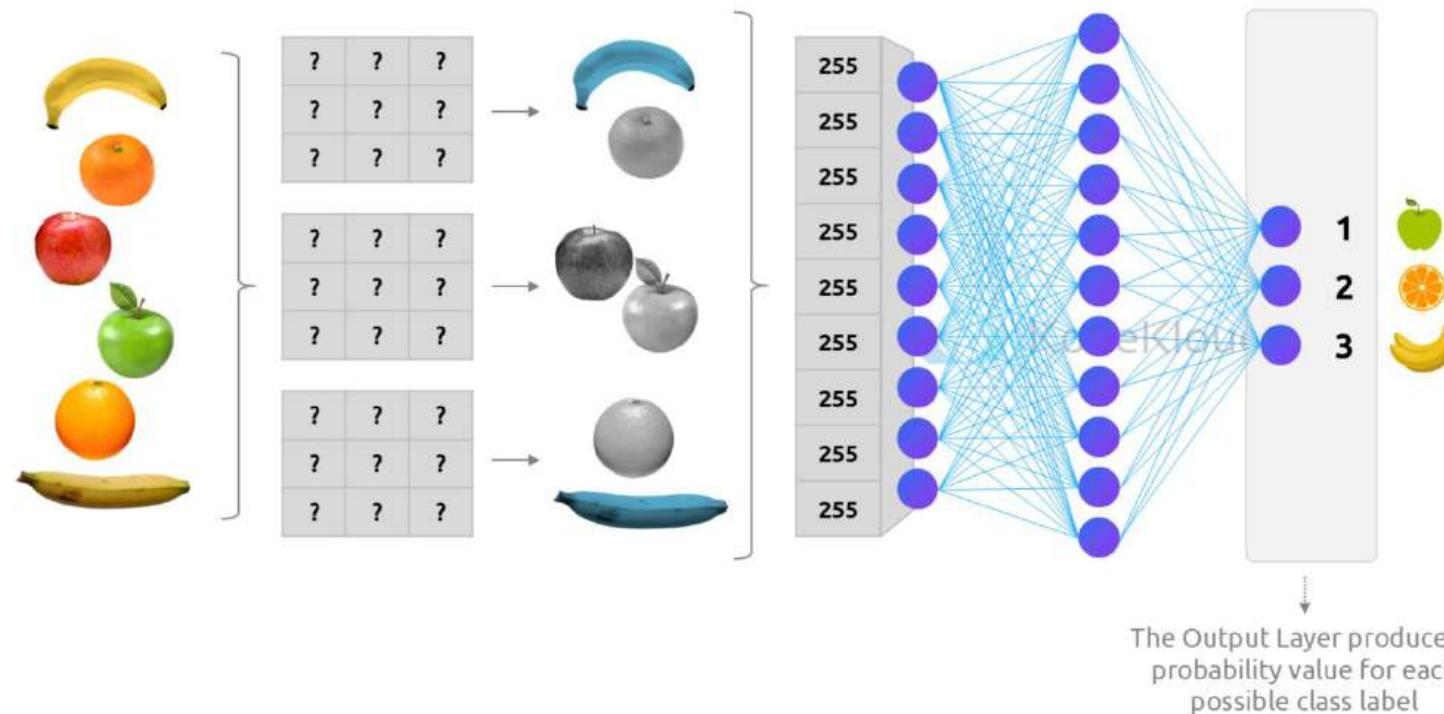
A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



© Copyright KodeKloud

A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks

During
Training

-1	-1	-1
-1	-8	-1
-1	-1	-1

Filter Kernels



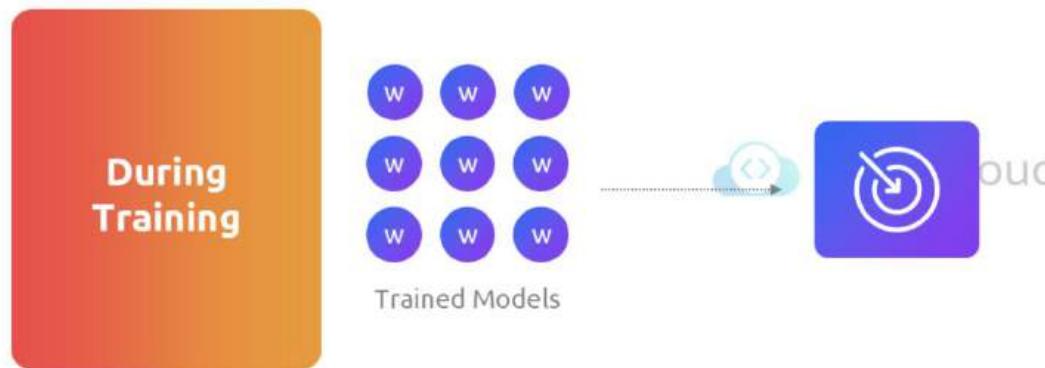
A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Convolutional Neural Networks



© Copyright KodeKloud

A grayscale image consists of an array of pixel values between 0 (black) and 255 (white) (a color picture consists of three arrays called channels, each containing pixel values for red, green, and blue hues)

Most image processing involves the application of filters, which consist of kernels that define an array of weight values. The filter kernel is convolved (passed across and down) over the image, calculating values for a new array by using a weighted sum of the original pixel values multiplied by the corresponding kernel weights.

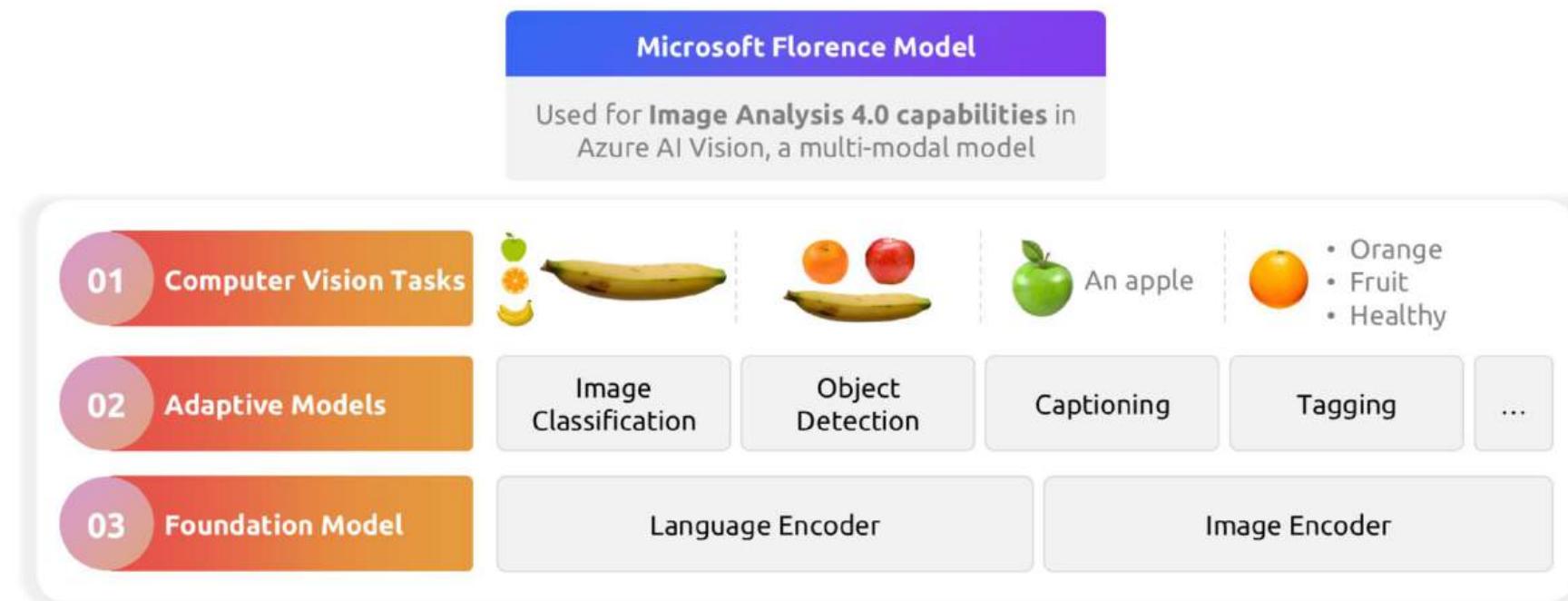
The new array represents the filtered image, usually with some kind of visual effect. In this case, a filter names a Laplace

filter has been used to highlight the edges of shapes in the original image.

Multi-Modal Models



Multi-Modal Models



© Copyright KodeKloud

The Microsoft Florence model, which is used for Image Analysis 4.0 capabilities in Azure AI Vision, is a multi-modal model. Trained with huge volumes of captioned images from the Internet, it includes both a language encoder and an image encoder. Florence is an example of a foundation model. In other words, a pre-trained general model on which you can build multiple adaptive models for specialist tasks. For example, you can use Florence as a foundation model for adaptive models that perform:

Image classification: Identifying to which category an image belongs.

Object detection: Locating individual objects within an image.

Captioning: Generating appropriate descriptions of images.

Tagging: Compiling a list of relevant text tags for an image.

Multi-Modal Models



Image Classification

Object Detection

Captioning

Tagging

Identifying the category to which an image belongs



The Microsoft Florence model, which is used for Image Analysis 4.0 capabilities in Azure AI Vision, is a multi-modal model. Trained with huge volumes of captioned images from the Internet, it includes both a language encoder and an image encoder. Florence is an example of a foundation model. In other words, a pre-trained general model on which you can build multiple adaptive models for specialist tasks. For example, you can use Florence as a foundation model for adaptive models that perform:

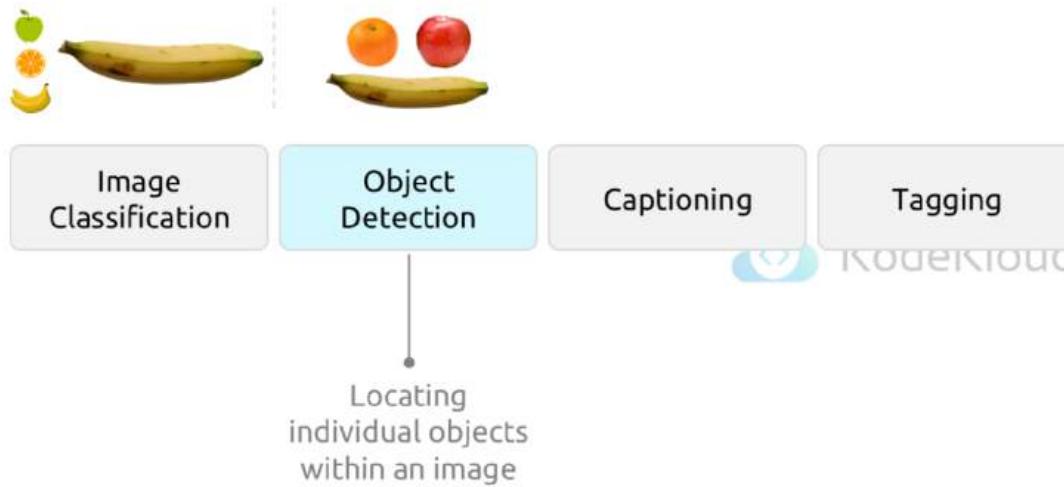
Image classification: Identifying to which category an image belongs.

Object detection: Locating individual objects within an image.

Captioning: Generating appropriate descriptions of images.

Tagging: Compiling a list of relevant text tags for an image.

Multi-Modal Models



© Copyright KodeKloud

The Microsoft Florence model, which is used for Image Analysis 4.0 capabilities in Azure AI Vision, is a multi-modal model. Trained with huge volumes of captioned images from the Internet, it includes both a language encoder and an image encoder. Florence is an example of a foundation model. In other words, a pre-trained general model on which you can build multiple adaptive models for specialist tasks. For example, you can use Florence as a foundation model for adaptive models that perform:

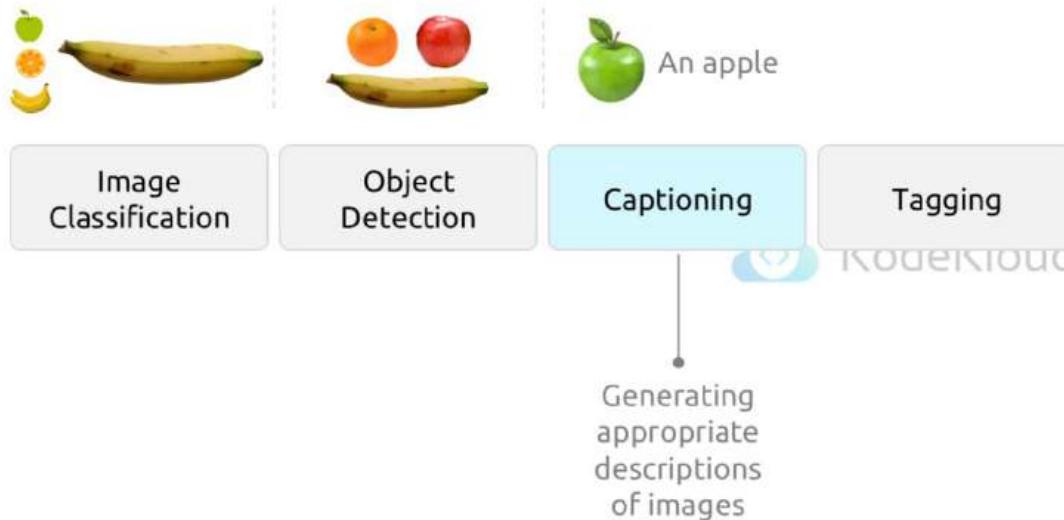
Image classification: Identifying to which category an image belongs.

Object detection: Locating individual objects within an image.

Captioning: Generating appropriate descriptions of images.

Tagging: Compiling a list of relevant text tags for an image.

Multi-Modal Models



© Copyright KodeKloud

The Microsoft Florence model, which is used for Image Analysis 4.0 capabilities in Azure AI Vision, is a multi-modal model. Trained with huge volumes of captioned images from the Internet, it includes both a language encoder and an image encoder. Florence is an example of a foundation model. In other words, a pre-trained general model on which you can build multiple adaptive models for specialist tasks. For example, you can use Florence as a foundation model for adaptive models that perform:

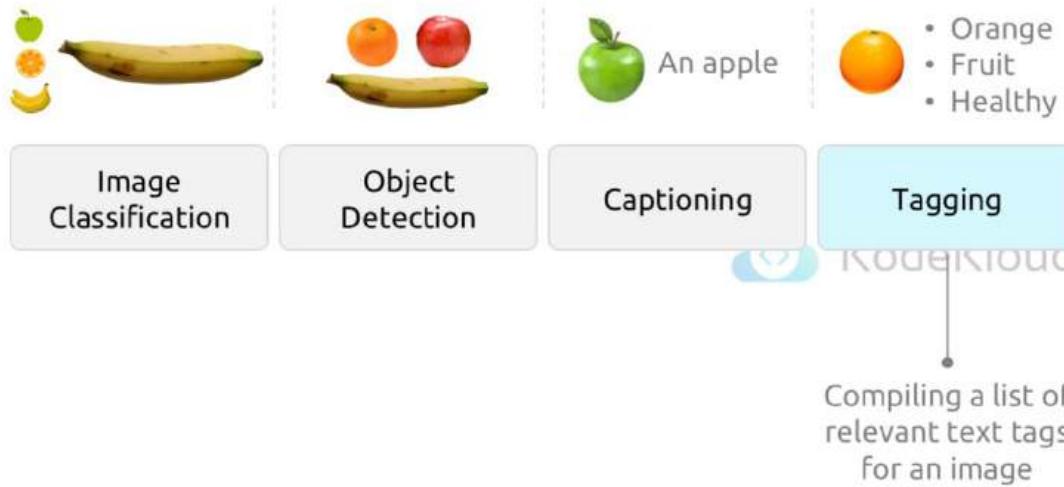
Image classification: Identifying to which category an image belongs.

Object detection: Locating individual objects within an image.

Captioning: Generating appropriate descriptions of images.

Tagging: Compiling a list of relevant text tags for an image.

Multi-Modal Models



© Copyright KodeKloud

The Microsoft Florence model, which is used for Image Analysis 4.0 capabilities in Azure AI Vision, is a multi-modal model. Trained with huge volumes of captioned images from the Internet, it includes both a language encoder and an image encoder. Florence is an example of a foundation model. In other words, a pre-trained general model on which you can build multiple adaptive models for specialist tasks. For example, you can use Florence as a foundation model for adaptive models that perform:

Image classification: Identifying to which category an image belongs.

Object detection: Locating individual objects within an image.

Captioning: Generating appropriate descriptions of images.

Tagging: Compiling a list of relevant text tags for an image.

Multi-Modal Models

Encapsulating semantic relationships



+



loud

Process image and
text data

Enhances the ability to understand and generate more comprehensive insights

© Copyright KodeKloud

Combining Language and Vision Models

A newer approach to modeling involves integrating language and vision models to simultaneously process image and text data. This fusion enhances the model's ability to understand and generate more comprehensive insights.

Multi-Modal Models

Encapsulating semantic relationships



Allows for more accurate interpretations and predictions across both modalities

© Copyright KodeKloud

Combining Language and Vision Models

A newer approach to modeling involves integrating language and vision models to simultaneously process image and text data. This fusion enhances the model's ability to understand and generate more comprehensive insights.

Multi-Modal Models

Foundation and adaptive models



Classification



Object Detection



Captioning



Tagging

A multi-modal model that is tailored to specific tasks

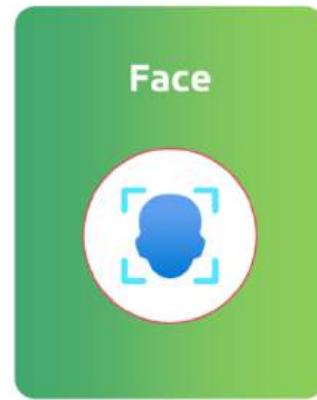
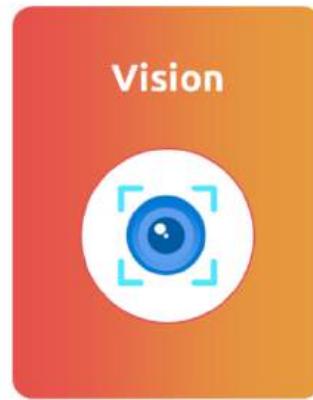
© Copyright KodeKloud

Foundation and Adaptive Models: A multi-modal model can serve as a foundation model, providing a robust base for more specialized adaptive models tailored to specific tasks such as classification, object detection, captioning, and tagging.

Azure Computer Vision Services

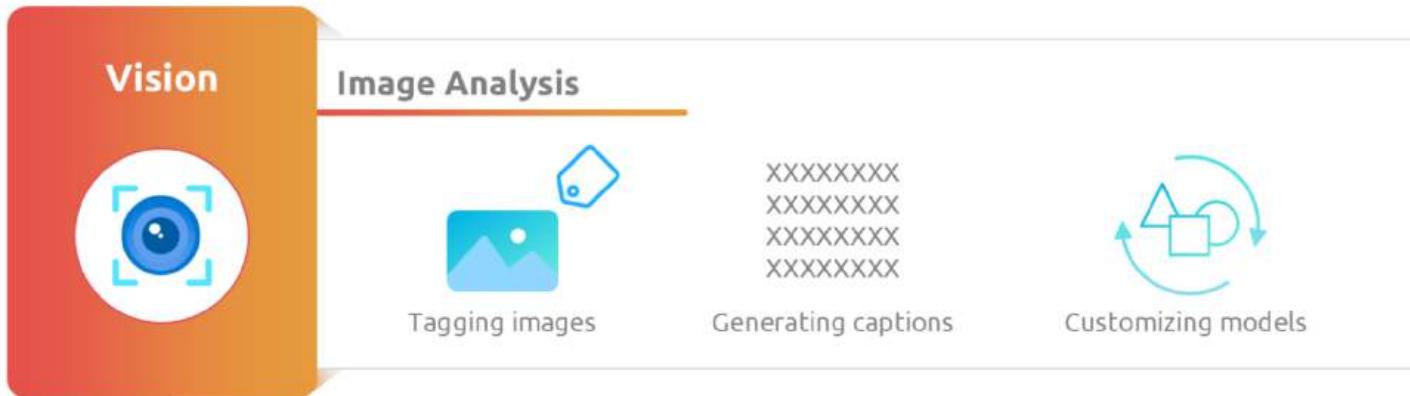


Azure Computer Vision Services



deKloud

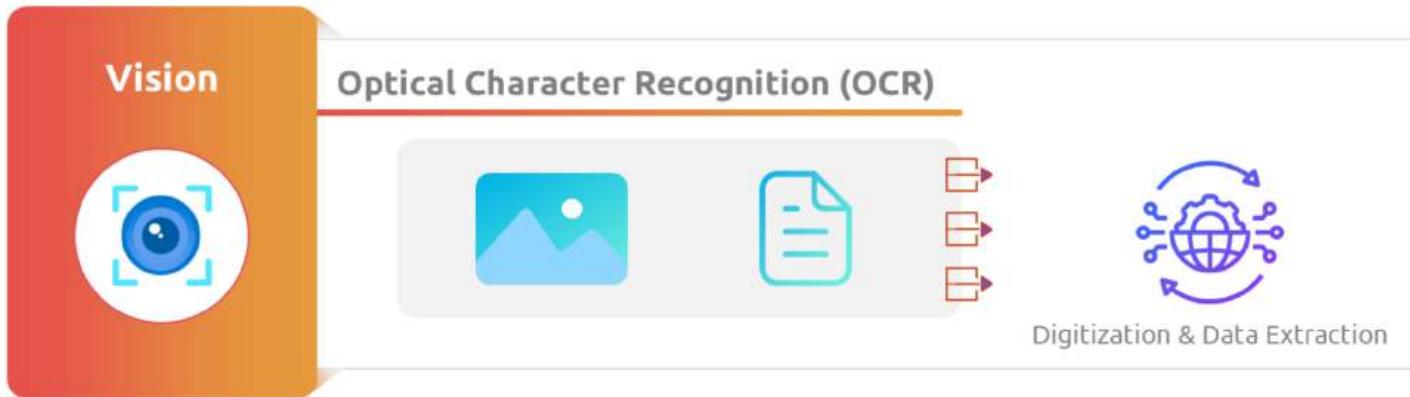
Azure Computer Vision Services



© Copyright KodeKloud

Image Analysis: Capabilities include tagging images, generating captions, and customizing models to suit specific needs.

Azure Computer Vision Services



© Copyright KodeKloud

Optical Character Recognition (OCR): Extracts text from images and documents, enabling digitization and data extraction

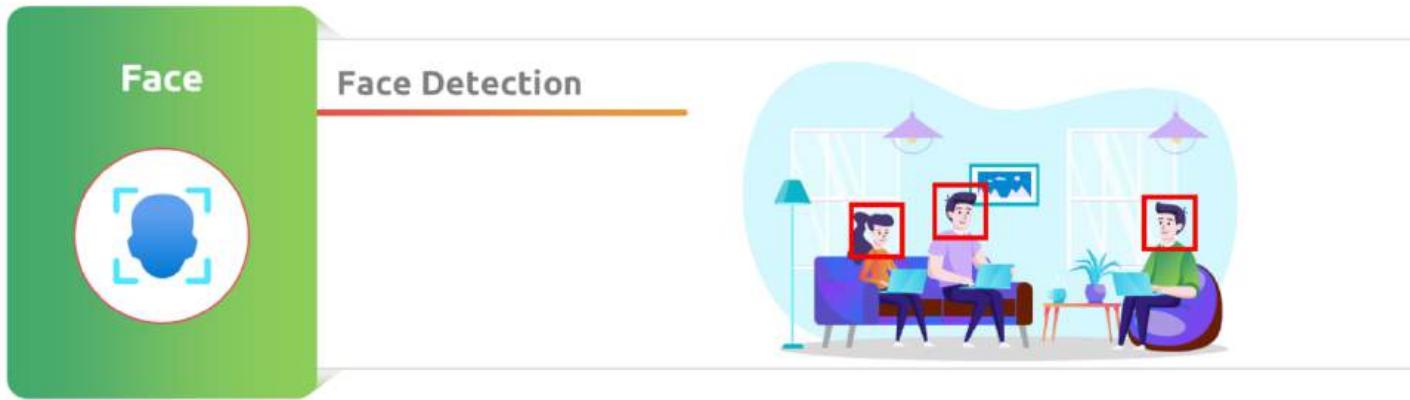
Azure Computer Vision Services



© Copyright KodeKloud

Spatial Analysis: Analyzes the spatial relationships within images, useful for tasks such as detecting object placement and movement.

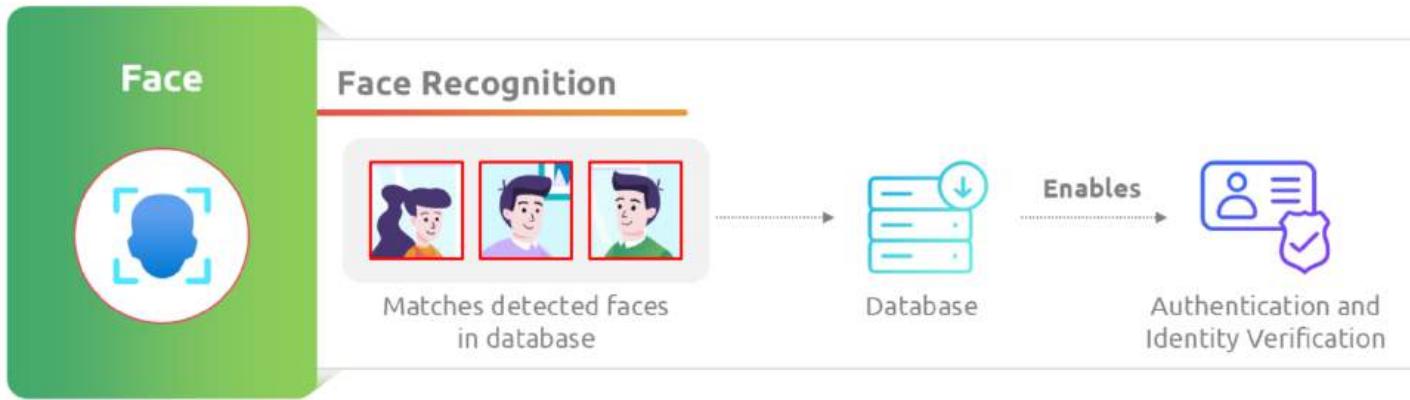
Azure Computer Vision Services



© Copyright KodeKloud

Face Detection: Identifies the presence of human faces in images, providing bounding boxes for each detected face.

Azure Computer Vision Services



© Copyright KodeKloud

Face Recognition: Matches detected faces to a database of known individuals, enabling applications like authentication and identity verification.

Azure Computer Vision Capabilities

Module Introduction

- 01 AI Vision Service and Image Analysis
- 02 Image Analysis Using Vision Studio
- 03 Face Service to Detect Faces
- 04 Detect Faces Using Face Service



KodeKloud

Module Introduction

05

Optical Character Recognition (OCR) to Read Text

06

Read Text Using OCR

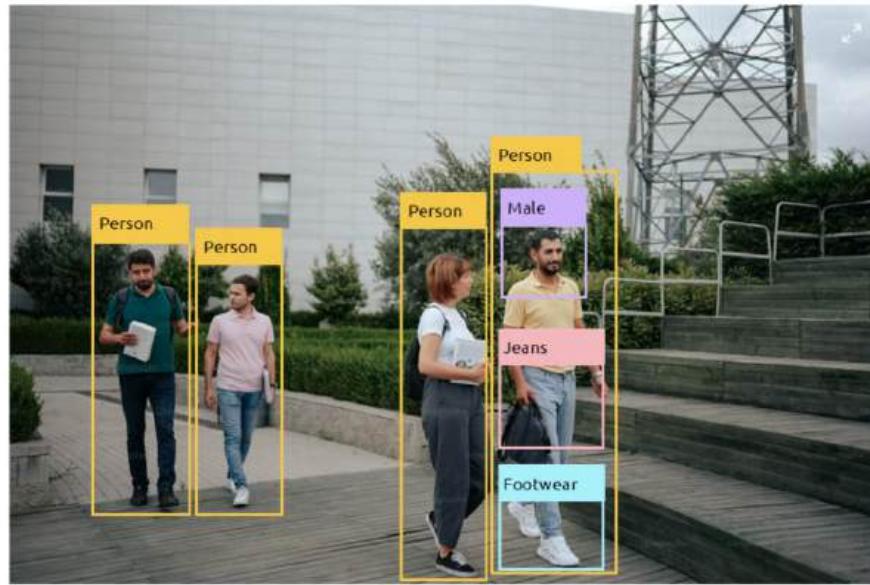


KodeKloud

AI Vision Service and Image Analysis



AI Vision Service and Image Analysis



Caption: A group of people walking on a sidewalk

Tags: Building, jeans, street, outdoor, jacket, city, person

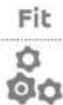
AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

MODEL CUSTOMIZATION

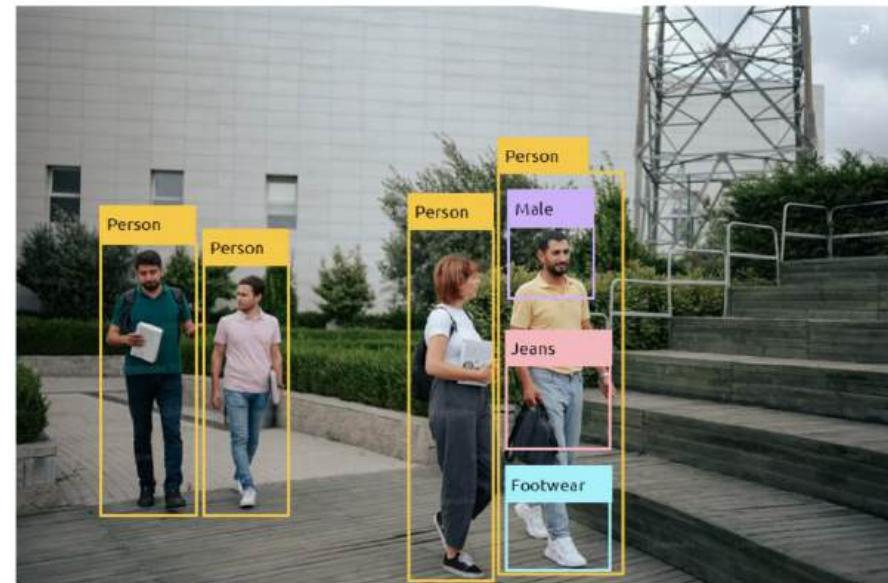


Tailor the image analysis models



Specific requirements

Improve accuracy for particular use cases



Caption: A group of people walking on a sidewalk

Tags: Building, jeans, street, outdoor, jacket, city, person

© Copyright KodeKloud

Model Customization: Tailor the image analysis models to fit specific requirements and improve accuracy for particular use cases.

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

READ TEXT FROM IMAGE

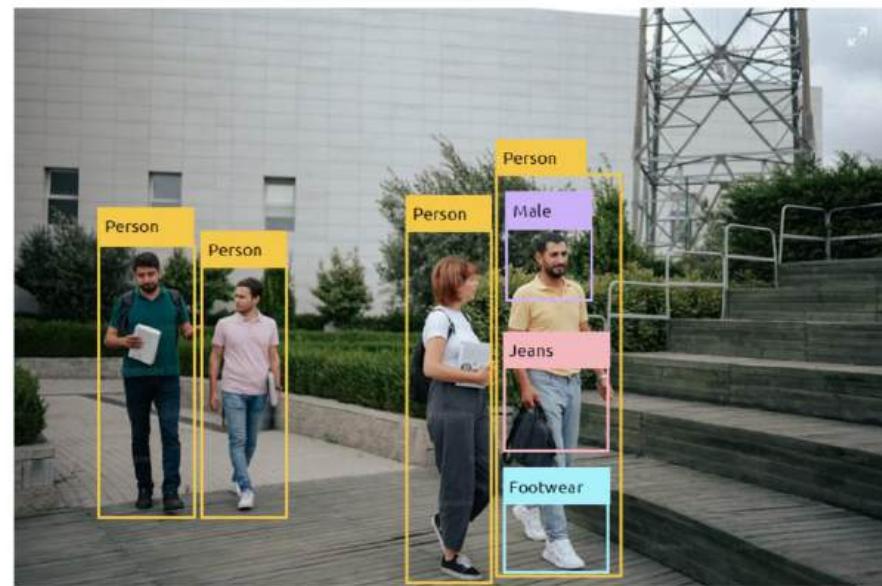


Extract & process text from image



Digitization of content

Enables digitization of printed and handwritten content



© Copyright KodeKloud

Read Text from Image: Extract and process text from images, enabling digitization of printed and handwritten content.

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

DETECT PEOPLE IN IMAGES

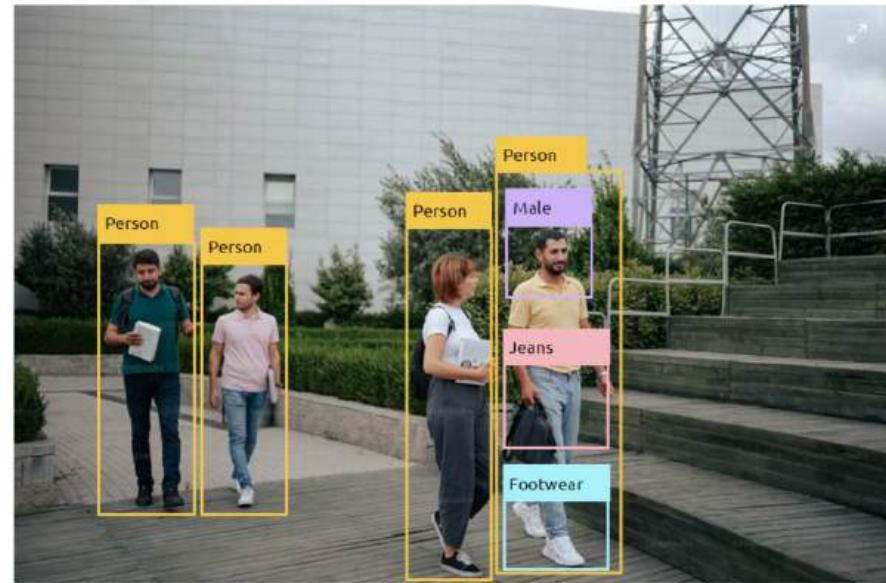


Identify
individuals in
image



Crowd
analysis

Useful for surveillance and crowd analysis



© Copyright KodeKloud

Detect People in Images: Identify and locate individuals within images, useful for surveillance and crowd analysis.

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

GENERATE IMAGE CAPTIONS

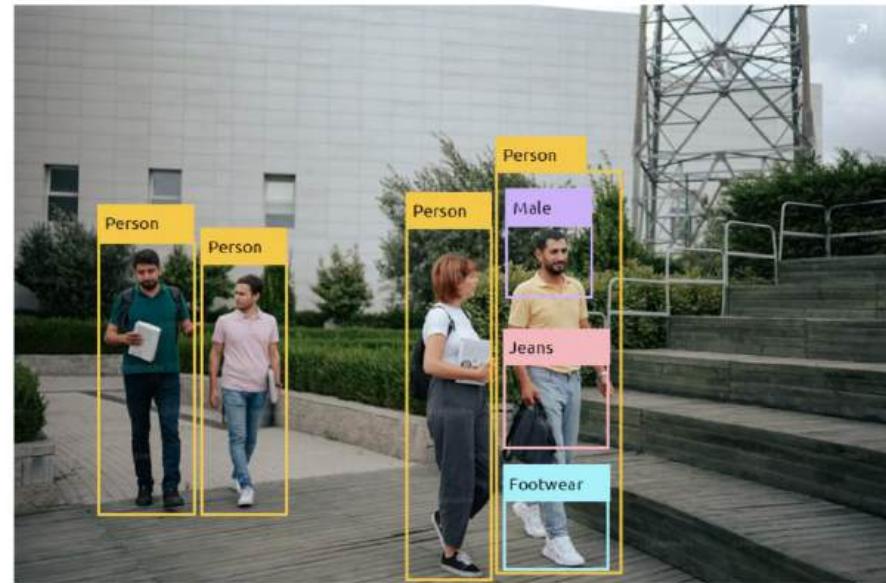


Captions for images



Content understanding

Enhances accessibility and content understanding



Caption: A group of people walking on a sidewalk

Tags: Building, jeans, street, outdoor, jacket, city, person

© Copyright KodeKloud

Generate Image Captions: Automatically produce descriptive captions for images, enhancing accessibility and content understanding.

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

DETECT OBJECTS

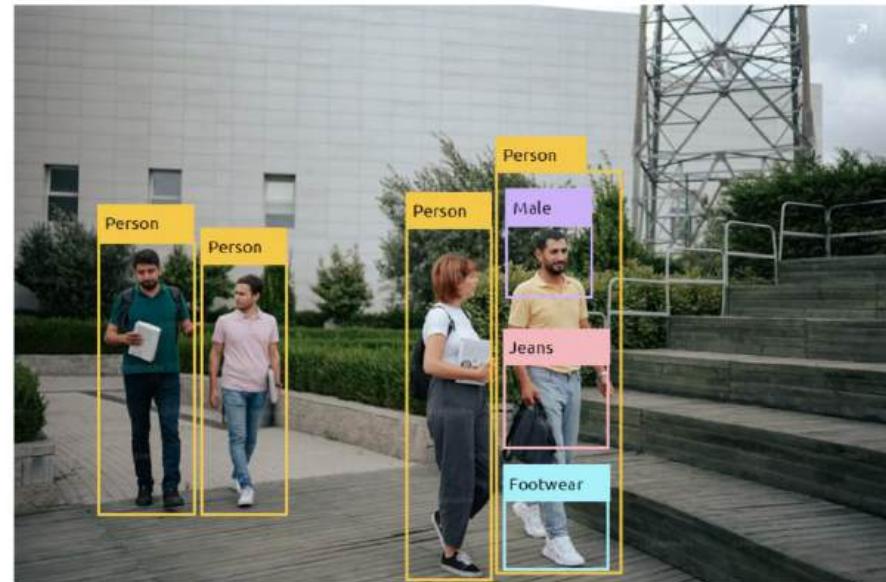


Label objects
in image



Supports
tasks

Supports tasks like inventory management and
autonomous navigation



© Copyright KodeKloud

Detect Objects: Recognize and label various objects within images, supporting tasks like inventory management and autonomous navigation

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

TAG VISUAL FEATURES

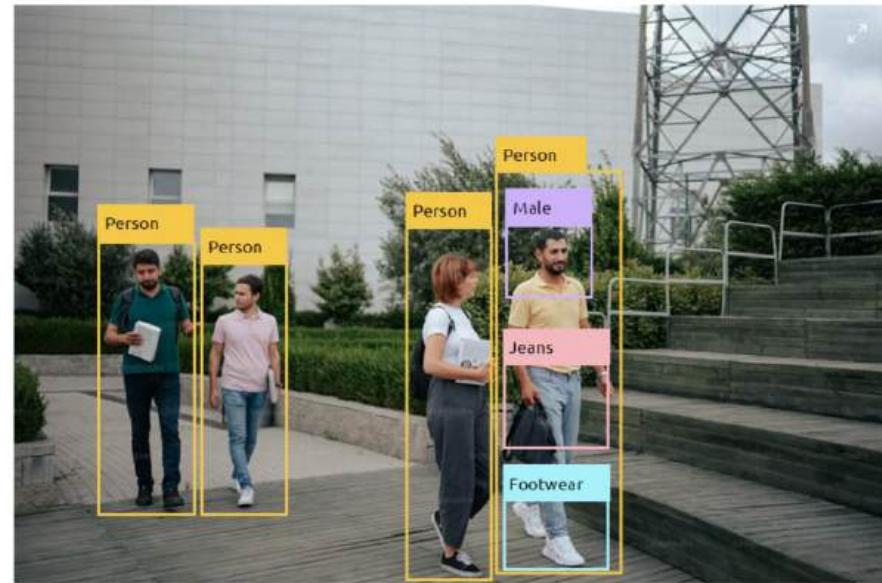


Tags of image



Categorization

Facilitates efficient search and categorization



Caption: A group of people walking on a sidewalk

Tags: Building, jeans, street, outdoor, jacket, city, person

© Copyright KodeKloud

Tag Visual Features: Assign descriptive tags to visual elements in images, facilitating efficient search and categorization

AI Vision Service and Image Analysis

CAPABILITIES INCLUDE

SMART CROP



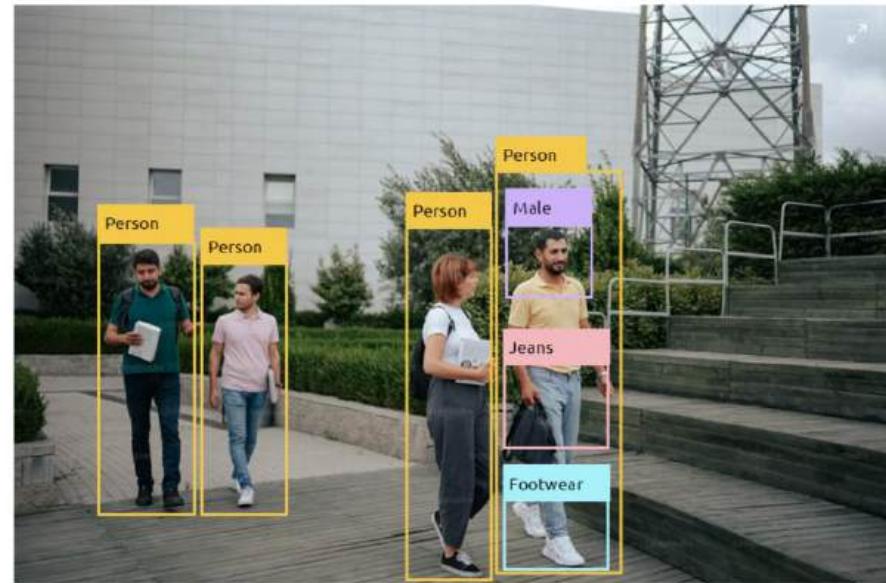
Crop images

Improves



Visual appeal

Improves visual appeal and focus



Caption: A group of people walking on a sidewalk

Tags: Building, jeans, street, outdoor, jacket, city, person

© Copyright KodeKloud

Smart Crop: Automatically crop images to highlight important content, improving visual appeal and focus

Face Service to Detect Faces



Face Service to Detect Faces

Anyone can use Face Service to Detect...



Identifies and assesses the level of blur in an image, ensuring the quality of face detection



Analyzes the exposure levels to ensure faces are well lit and detectable



Detects the presence of glasses on a face, which can be crucial for identification and analysis



Determines the orientation of the head, including pitch, yaw, and roll angles



Evaluates the image noise that may affect the clarity of face detection



Identifies if the face is partially covered by objects like masks or hands, which may hinder detection



© Copyright KodeKloud

Anyone can use the Face service to detect:

Blur: Identifies and assesses the level of blur in an image, ensuring the quality of face detection.

Exposure: Analyzes the exposure levels to ensure faces are well-lit and detectable.

Glasses: Detects the presence of glasses on a face, which can be crucial for identification and analysis.

Head Pose: Determines the orientation of the head, including pitch, yaw, and roll angles.

Noise: Evaluates the image noise that may affect the clarity of face detection.

Occlusion: Identifies if the face is partially covered by objects like masks or hands, which may hinder detection.

Face Service to Detect Faces

Only Managed Microsoft Customers can access
Facial Recognition Capabilities

Similarity Matching

- Compares faces to determine if they belong to the same person
- Useful for tasks like deduplication and finding similar faces

Identity Verification

- Verifies the identity of a person by comparing their face to a known face in a database
- Crucial for security and access control



© Copyright KodeKloud

Only Managed Microsoft customers can access facial recognition capabilities:

Similarity Matching: Compares faces to determine if they belong to the same person, useful for tasks like deduplication and finding similar faces.

Identity Verification: Verifies the identity of a person by comparing their face to a known face in a database, crucial for security and access control.

Optical Character Recognition (OCR) to Read Text



Optical Character Recognition (OCR) to Read Text

Detect the Location of Text



Printed

Identifies and extracts printed text from various types of images, including scanned documents, photographs, and digital images



Handwritten

Recognizes and captures handwritten text, enabling the digitization of notes, forms, and other handwritten materials



© Copyright KodeKloud

Detect the Location of Text:

Printed: Identifies and extracts printed text from various types of images, including scanned documents, photographs, and digital images.

Handwritten: Recognizes and captures handwritten text, enabling the digitization of notes, forms, and other handwritten materials.

Optical Character Recognition (OCR) to Read Text

OPTIONS FOR QUICK TEXT EXTRACTION



From Images

Enables rapid extraction of text from images, making it easy to convert visual data into editable and searchable text



Asynchronous Analysis of Larger Scanned Documents

Processes large volumes of scanned documents asynchronously, allowing for efficient and scalable text extraction without real-time constraints



© Copyright KodeKloud

Options for Quick Text Extraction:

From Images: Enables rapid extraction of text from images, making it easy to convert visual data into editable and searchable text.

Asynchronous Analysis of Larger Scanned Documents: Processes large volumes of scanned documents asynchronously, allowing for efficient and scalable text extraction without real-time constraints.

Natural Language Processing (NLP)

Module Introduction

01

Introduction to Natural Language Processing (NLP)



KodeKloud

02

NLP and Conversational AI in Azure

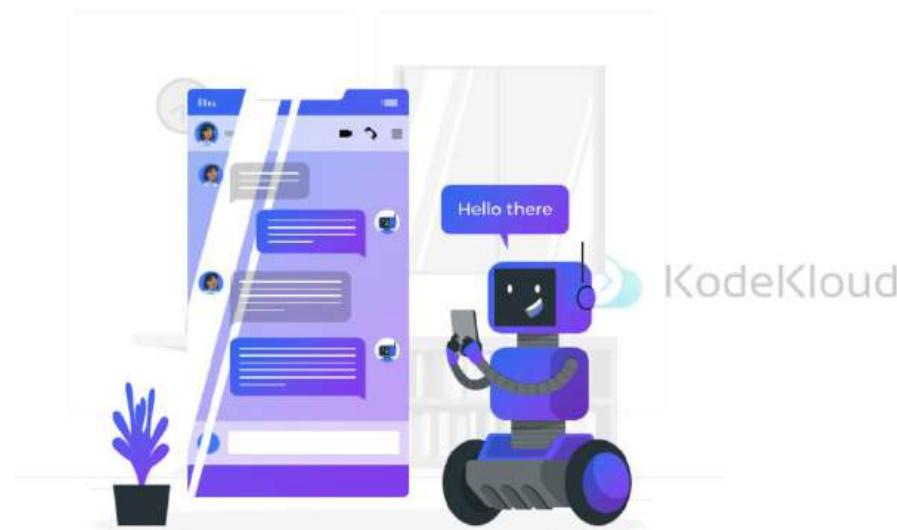
Natural Language Processing – Introduction



© Copyright KodeKloud

Introduction to Natural Language Processing

Natural Language Processing – Introduction



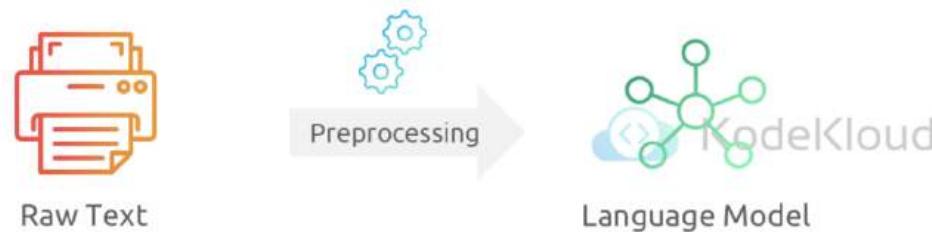
KodeKloud

Interprets written and spoken language

© Copyright KodeKloud

Natural language processing (NLP) is the area of AI that deals with making sense of written and spoken language.

Natural Language Processing – Introduction



© Copyright KodeKloud

Typically, an NLP solution starts with a large volume of raw text, which is then pre-processed to prepare it for modeling.

Natural Language Processing – Introduction

Preprocessing techniques:

1. Stop-word removal

Eliminating common words
with little semantic value,
like "a", "the", etc.



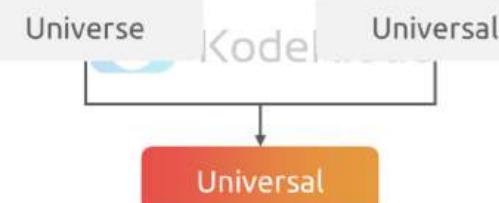
There are many pre-processing techniques that can be applied to text, including stop-word removal (eliminating common words with little semantic value, like "a", "the", etc.),

Natural Language Processing – Introduction

Preprocessing techniques:

2. Stemming or lemmatization

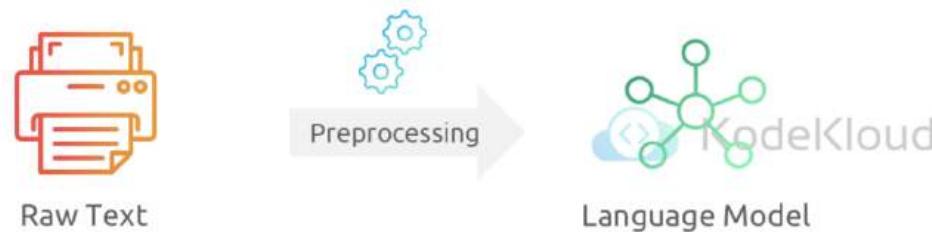
Coalescing words with the same root as a single word



© Copyright KodeKloud

stemming or lemmatization (coalescing words with the same root as a single word – so for example, “universe” and “universal” might both be represented as “univers”), and other techniques. Many of these approaches were extensively used in statistical modeling techniques that relied on counting the frequency of words in text to derive meaning, but are not so important in modern deep learning techniques.

Natural Language Processing – Introduction



© Copyright KodeKloud

One thing that is commonly done though is to tokenize the words – representing each distinct word or phrase with a unique numeric identifier. The set of tokens for a given corpus of text represents the vocabulary for a language model. The data is then used to train a model.

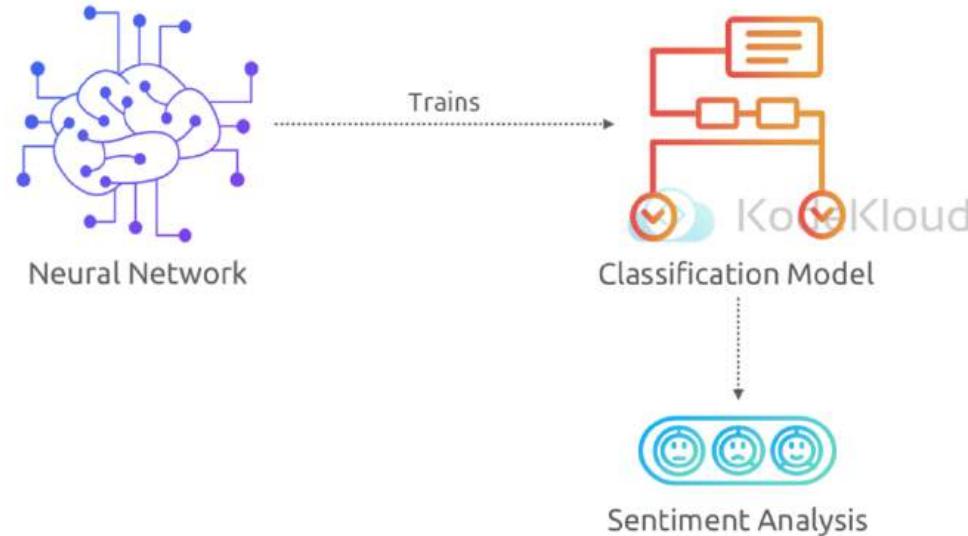
Natural Language Processing – Introduction



© Copyright KodeKloud

One thing that is commonly done though is to tokenize the words – representing each distinct word or phrase with a unique numeric identifier. The set of tokens for a given corpus of text represents the vocabulary for a language model. The data is then used to train a model. Depending on the type of task you want to accomplish, the details of the model will vary.

Natural Language Processing – Introduction



© Copyright KodeKloud

For example, you might use a fully connected neural network to train a classification model for sentiment analysis (predicting whether a body of text is “positive” or “negative”)

Natural Language Processing – Introduction



KodeKloud



Positive

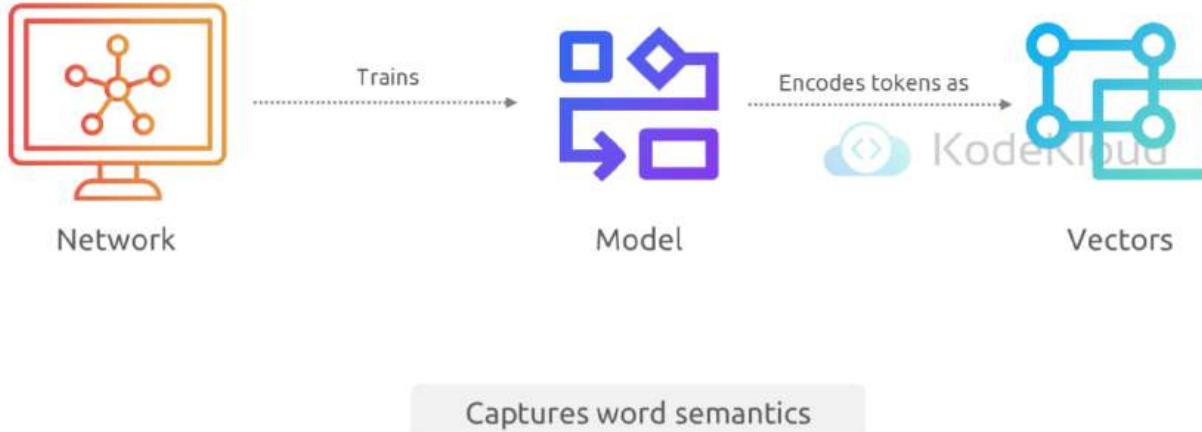


Negative

© Copyright KodeKloud

For example, you might use a fully-connected neural network to train a classification model for sentiment analysis (predicting whether a body of text is “positive” or “negative”)

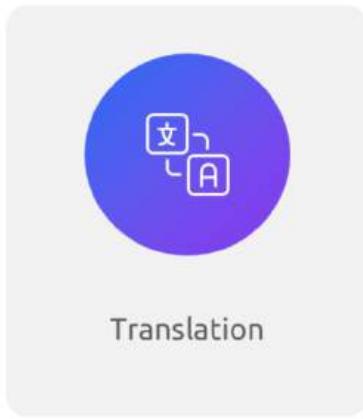
Natural Language Processing – Introduction



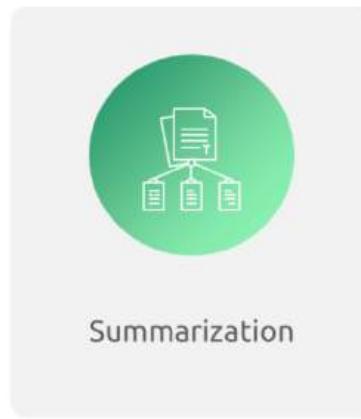
© Copyright KodeKloud

or you might use a more sophisticated network to train a model that encodes language tokens as vectors that encapsulate semantic relationships between words in the vocabulary.

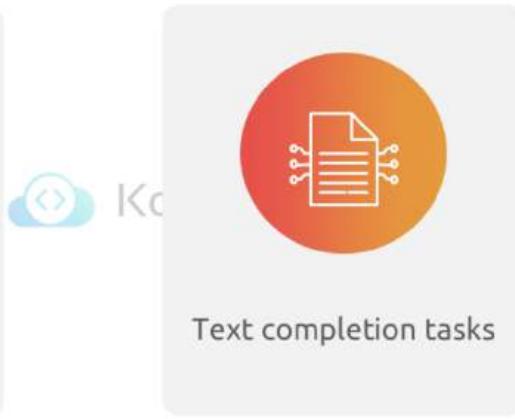
Natural Language Processing – Introduction



Translation



Summarization



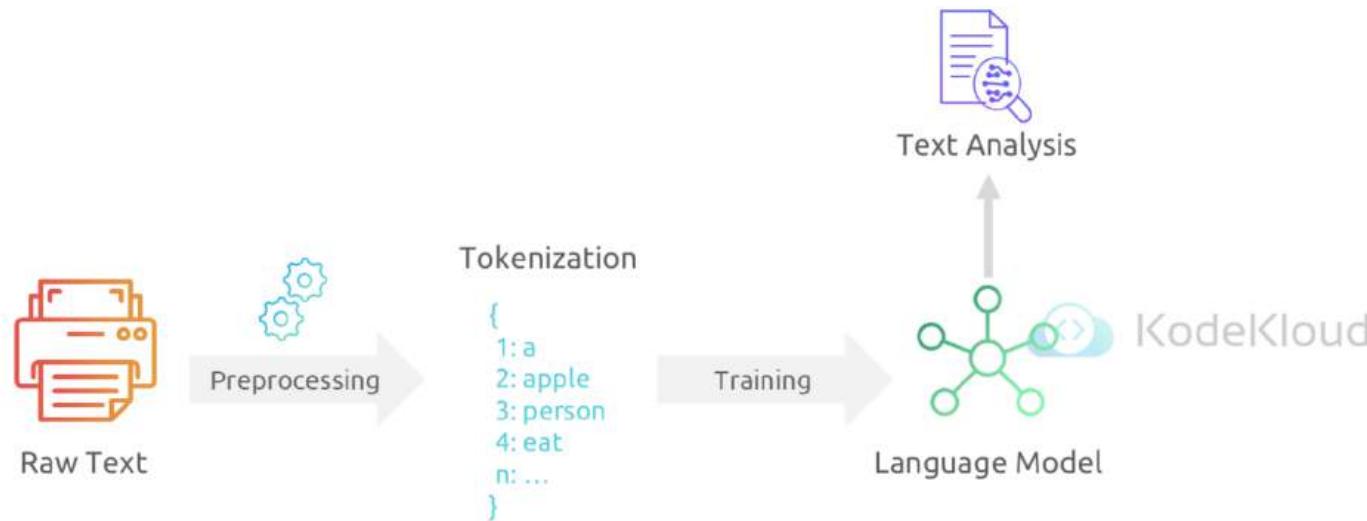
Text completion tasks

Such models are useful for translation, summarization, and text completion tasks.

Natural Language Processing – Introduction



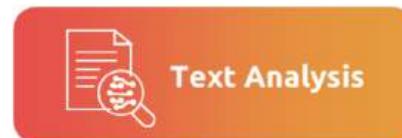
Natural Language Processing – Introduction



© Copyright KodeKloud

Common tasks that a language model can support include:
Text analysis
Opinion mining
Machine translation
Summarization
Conversational AI

Natural Language Processing – Introduction



KodeKloud

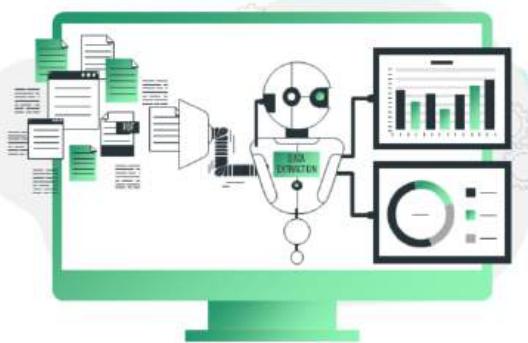
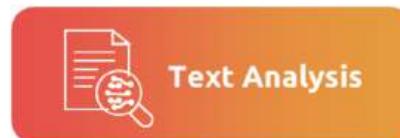
Extract phrases

Identify entities

© Copyright KodeKloud

Text analysis– Often you need to analyze a text document to extract key phrases or to identify entities it mentions, such as dates, places, people. For example, a company might use AI to analyze industry magazine articles to try to find articles that mention their products or executives or to determine the main subject of each article.

Natural Language Processing – Introduction

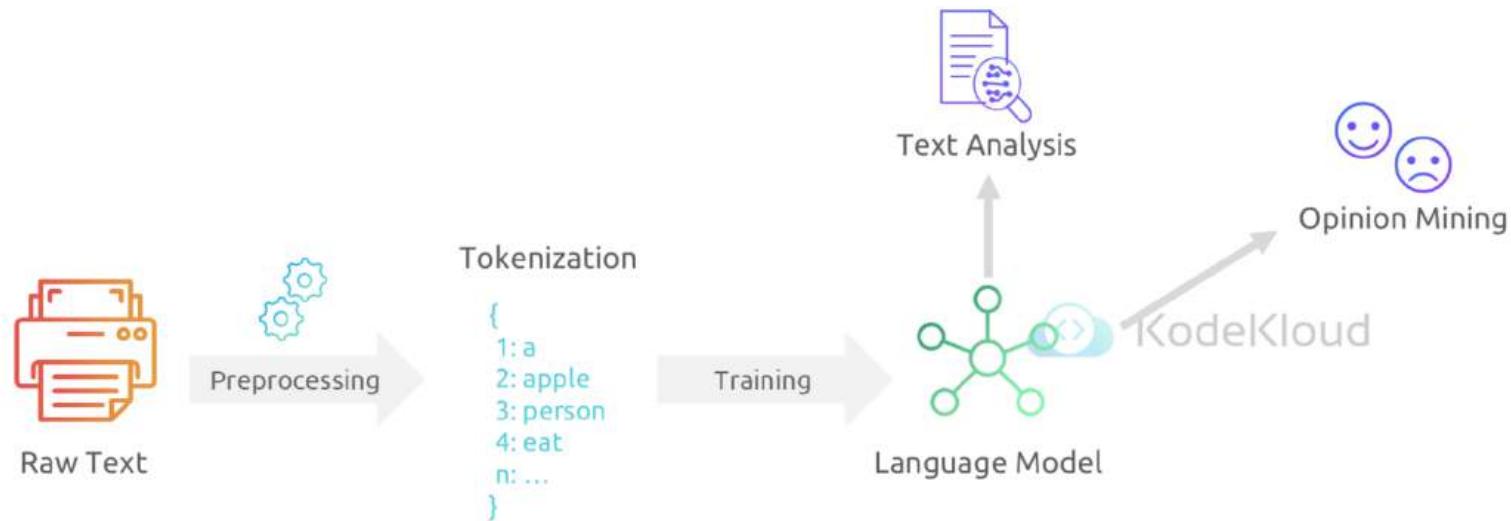


KodeKloud

© Copyright KodeKloud

Text analysis– Often you need to analyze a text document to extract key phrases or to identify entities it mentions, such as dates, places, people. For example, a company might use AI to analyze industry magazine articles to try to find articles that mention their products or executives or to determine the main subject of each article.

Natural Language Processing – Introduction



© Copyright KodeKloud

Common tasks that a language model can support include:
Text analysis
Opinion mining
Machine translation
Summarization
Conversational AI

Natural Language Processing – Introduction



© Copyright KodeKloud

Opinion mining and Sentiment analysis – This is a common form of text analysis that calculates a score indicating how positive (or negative) a text extract is. For example, a retailer might analyze reviews from customers to determine which ones are positive and which are negative and analyze the text to uncover the key drivers of sentiment.

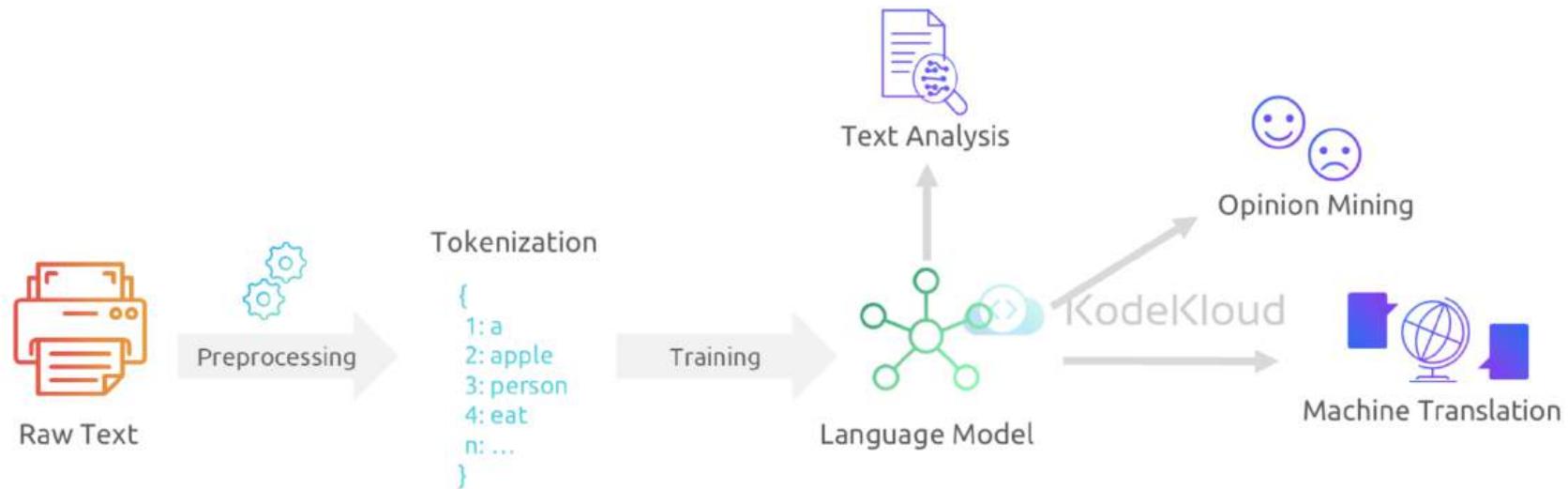
Natural Language Processing – Introduction



© Copyright KodeKloud

Opinion mining and Sentiment analysis – This is a common form of text analysis that calculates a score indicating how positive (or negative) a text extract is. For example, a retailer might analyze reviews from customers to determine which ones are positive and which are negative and analyze the text to uncover the key drivers of sentiment.

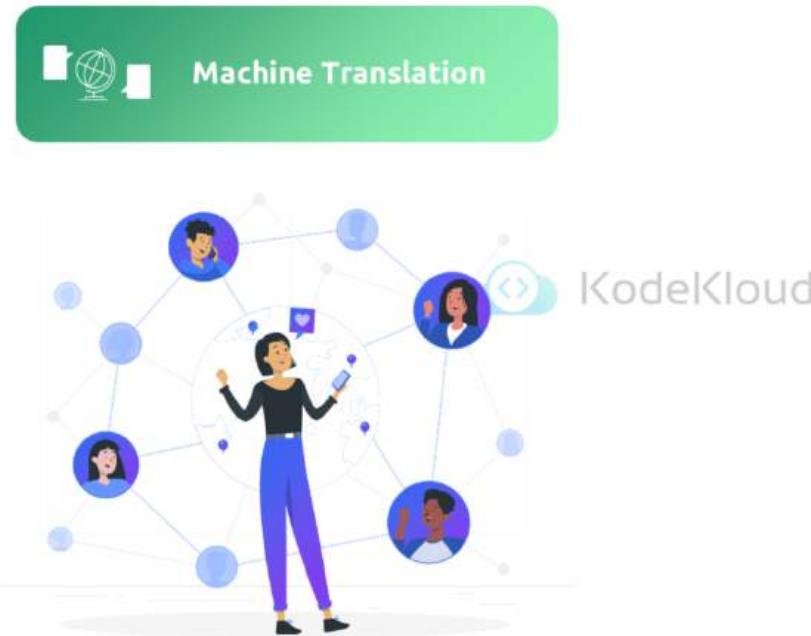
Natural Language Processing – Introduction



© Copyright KodeKloud

Common tasks that a language model can support include:
Text analysis
Opinion mining
Machine translation
Summarization
Conversational AI

Natural Language Processing – Introduction



© Copyright KodeKloud

Machine translation – International and cross-cultural collaboration is often a key to success, and this requires the ability to eliminate language barriers.

Natural Language Processing – Introduction



KodeKloud

© Copyright KodeKloud

AI can be used to automate translation of written and spoken language. For example, an inbox add-in might be used to automatically translate incoming or outgoing emails, or a conference call presentation system might provide a simultaneous transcript of the speaker's words in multiple languages.

Natural Language Processing – Introduction



An inbox add-in can automatically translate emails.

© Copyright KodeKloud

For example, an inbox add-in might be used to automatically translate incoming or outgoing emails, or a conference call presentation system might provide a simultaneous transcript of the speaker's words in multiple languages.

Natural Language Processing – Introduction



Machine Translation



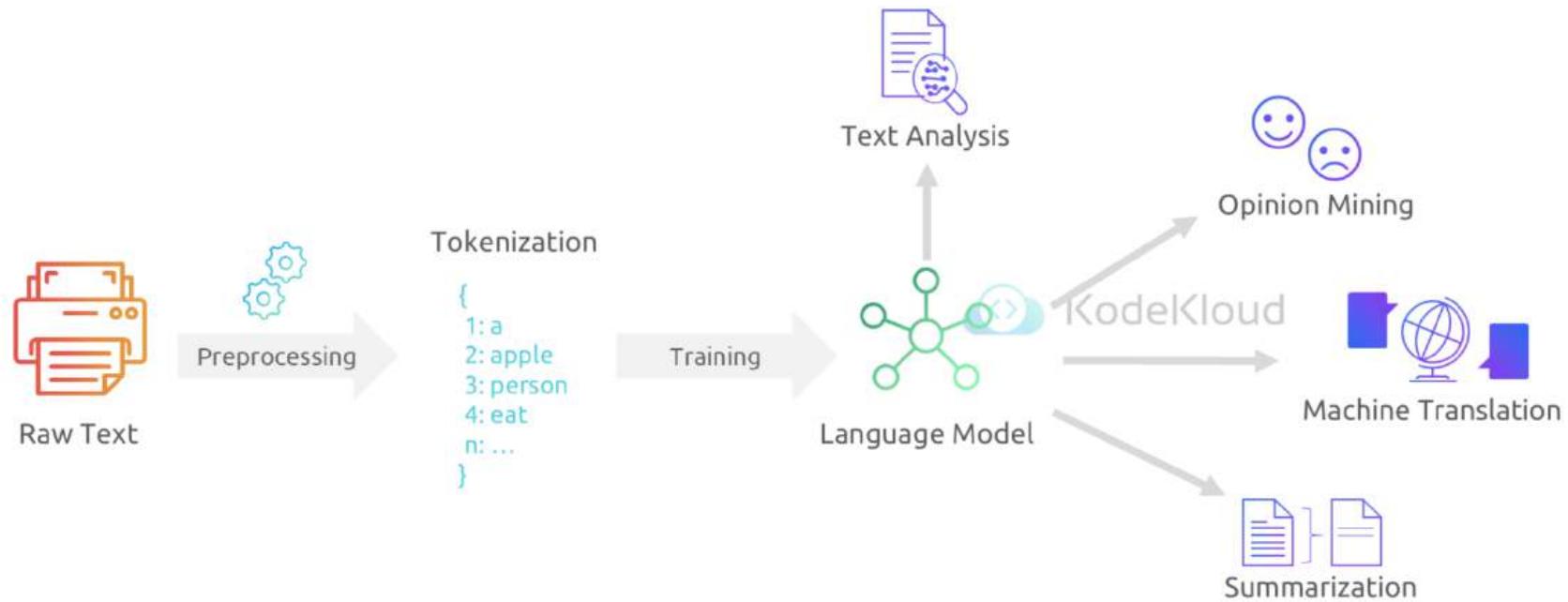
KodeKloud

A conference call system provides multilingual transcripts.

© Copyright KodeKloud

For example, an inbox add-in might be used to automatically translate incoming or outgoing emails, or a conference call presentation system might provide a simultaneous transcript of the speaker's words in multiple languages.

Natural Language Processing – Introduction



© Copyright KodeKloud

Common tasks that a language model can support include:
Text analysis
Opinion mining
Machine translation
Summarization
Conversational AI

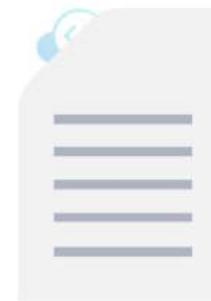
Natural Language Processing – Introduction



Summarization



Large volume of text

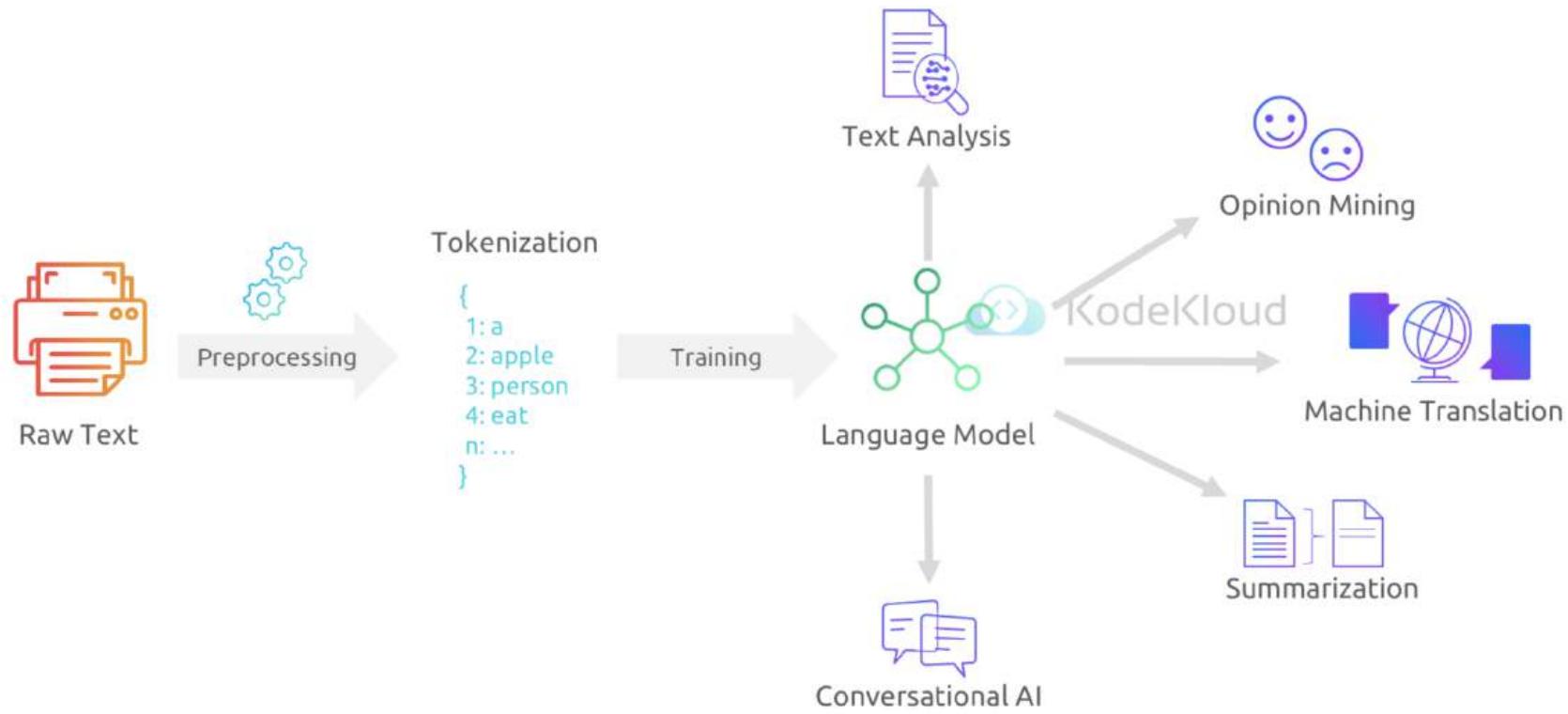


eKloud

Key points

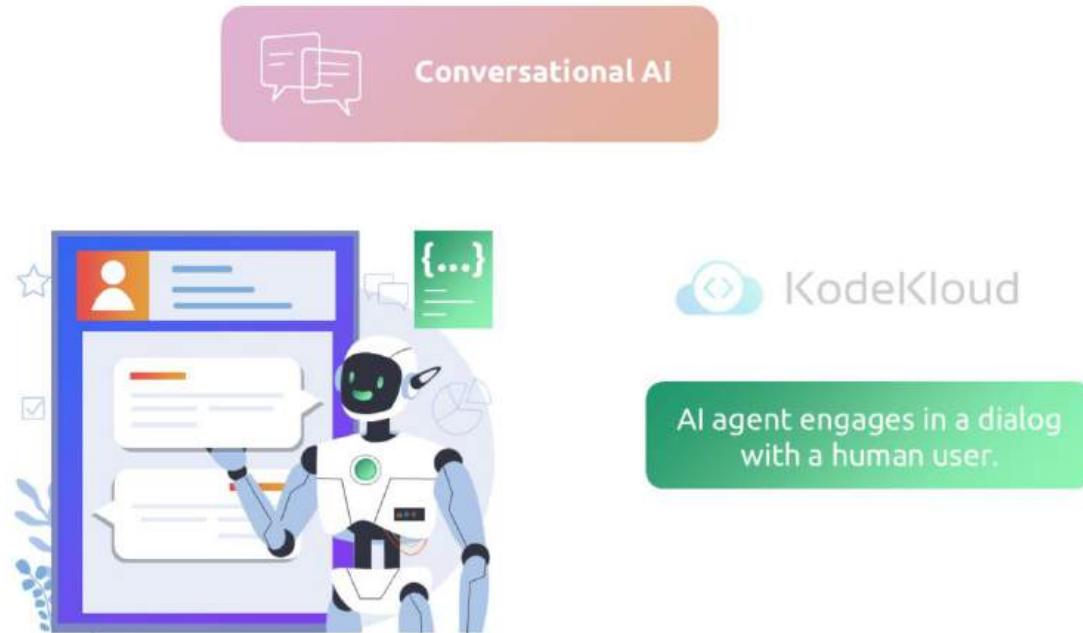
Summarization: Commonly, you might want to take a large volume of text and generate a summary of the key points.

Natural Language Processing – Introduction



Common tasks that a language model can support include:
Text analysis
Opinion mining
Machine translation
Summarization
Conversational AI

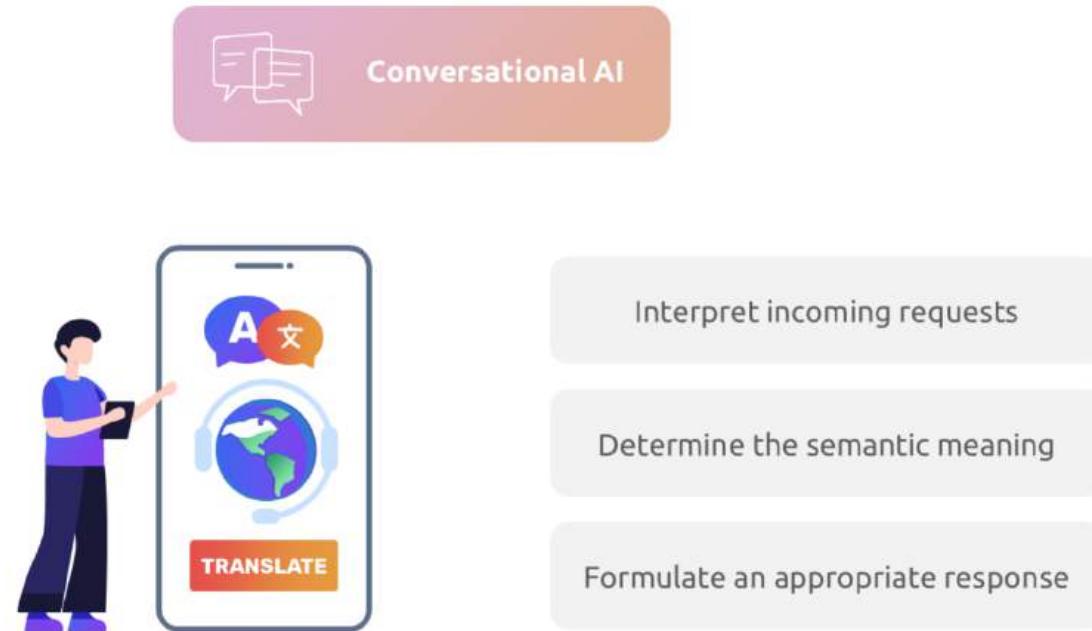
Natural Language Processing – Introduction



© Copyright KodeKloud

Conversational AI– Conversational AI is a form of AI in which an AI agent engages in a dialog with a human user (commonly, this form of application is described as a “bot”) To support these kinds of interactions, a language model must be developed that enables the AI agent to interpret incoming requests, determine the semantic meaning or “intent” of the request, and formulating an appropriate response.

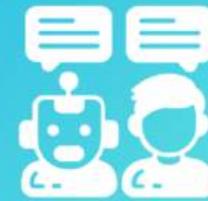
Natural Language Processing – Introduction



© Copyright KodeKloud

Conversational AI – Conversational AI is a form of AI in which an AI agent engages in a dialog with a human user (commonly, this form of application is described as a “bot”). To support these kinds of interactions, a language model must be developed that enables the AI agent to interpret incoming requests, determine the semantic meaning or “intent” of the request, and formulating an appropriate response.

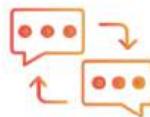
NLP and Conversational AI in Azure



© Copyright KodeKloud

NLP and Conversational AI in Azure

Convolutional Neural Networks



Language

- Language detection
- Key phrase extraction
- Named entity detection
- Sentiment analysis and opinion mining
- Personal information detection
- Summarization
- Conversational language understanding

eKloud

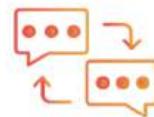
© Copyright KodeKloud

We'll cover some of the key capabilities of these NLP related services in the rest of this module.

For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Convolutional Neural Networks



Language

- Language detection
- Key phrase extraction
- Named entity detection
- Sentiment analysis and opinion mining
- Personal information detection
- Summarization
- Conversational language understanding



Language service

© Copyright KodeKloud

We'll cover some of the key capabilities of these NLP related services in the rest of this module.

For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Convolutional Neural Networks



- Text to speech
- Speech to text
- Speech translation
- Speaker identification
- Language identification

KodeKloud

© Copyright KodeKloud

We'll cover some of the key capabilities of these NLP related services in the rest of this module.

For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Convolutional Neural Networks



- Text to speech
- Speech to text
- Speech translation
- Speaker identification
- Language identification



[Speech service](#)

© Copyright KodeKloud

We'll cover some of the key capabilities of these NLP related services in the rest of this module.

For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Convolutional Neural Networks



- Text translation
- Document translation
- Custom translation

KodeKloud

© Copyright KodeKloud

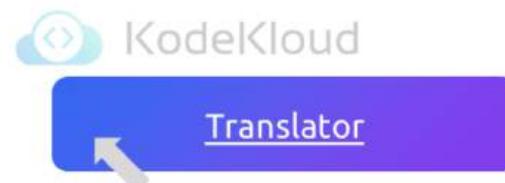
We'll cover some of the key capabilities of these NLP related services in the rest of this module.
For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Convolutional Neural Networks



- Text translation
- Document translation
- Custom translation



© Copyright KodeKloud

We'll cover some of the key capabilities of these NLP related services in the rest of this module.

For more details of the capabilities of these AI services, see the documentation at:

- <https://learn.microsoft.com/azure/ai-services/language-service/>
- <https://learn.microsoft.com/azure/ai-services/speech-service/>
- <https://learn.microsoft.com/azure/ai-services/translator/>

Azure NLP Services

Module Introduction

01 Text Analysis

02 Text Analysis Using Language Studio (Lab)



03 Question Answering

04 Azure Bot Service

Text Analysis

Text Analysis using Language Studio (Lab)

Question Answering

Azure Bot Service

Module Introduction

- 05 Q&A Model Using Language Studio (Lab)
- 06 Conversational Language and Understanding
- 07 Conversational Language and Understanding With Language Studio (Lab)
- 08 Speech Recognition and Synthesis



Conversational Language and Understanding

Conversational Language and Understanding with Language Studio (Lab)

Speech Recognition and Synthesis

QnA Model using Language Studio (Lab)

Module Introduction

09

Using Speech Studio (Lab)



KodeKloud

© Copyright KodeKloud

Using Speech Studio (Lab)

Text Analysis



Text Analysis

Azure AI | Language Studio

Welcome to Language Studio

Recent custom projects you've worked on

+ Create new ▾

Name ▾	Type ▾
[Project Name]	[Type]

[View all projects](#)

☆ Featured Extract information Classify text Understand questions and conversational language

Analyze sentiment and mine opinions
Detect positive, negative and neutral sentiment in text. Get more insights by mining opinions.

[Try it out](#)

Detect language
Evaluate text and detect a wide range of languages and variant dialects.

[Try it out](#)

Custom text classification
Train a classification model to classify text using your own data.

[Open Custom text classification](#)

Text Analysis



KodeKloud

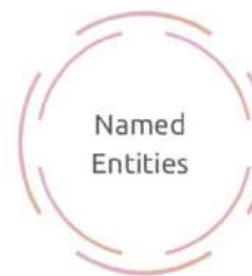
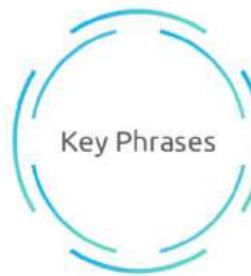
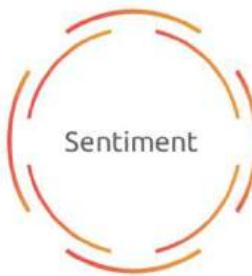
© Copyright KodeKloud

Azure's text analytics services can quickly analyze text to identify the predominant language, sentiment, key phrases, and named entities.

Text Analysis



KodeKloud



© Copyright KodeKloud

Azure's text analytics services can quickly analyze text to identify the predominant language, sentiment, key phrases, and named entities.

Text Analysis

I enjoyed a great meal in Italy.

- **Predominant Language:** English
- **Sentiment:** 0.92 (positive)
- **Key Phrases:** "great meal"
- **Entities:** Italy



KodeKloud

© Copyright KodeKloud

In this example, the sentence "I enjoyed a great meal in Italy" is analyzed to determine that it is written in English, expresses a positive sentiment with a high score, highlights the key phrase "great meal," and identifies "Italy" as a named entity.

Text Analysis – Importance



© Copyright KodeKloud

This functionality is useful for processing large volumes of text data, gaining insights into customer feedback, and extracting valuable information for further analysis.

Question Answering

By entering questions and answers:

Manually enter Q&A pairs to build a custom knowledge base



By using Built-In Chit-Chat: Use pre-built responses for common small talk



From an existing FAQ Document: Import FAQs to populate the knowledge base

© Copyright KodeKloud

Define a knowledge base of question-and-answer pairs:

By Entering Questions and Answers: Manually input questions and their corresponding answers to create a customized knowledge base.

From an Existing FAQ Document: Import frequently asked questions and answers from existing documents to quickly populate the knowledge base.

By Using Built-In Chit-Chat: Utilize pre-built conversational responses to handle common small talk and casual interactions.

Consume the knowledge base from client apps, including bots.

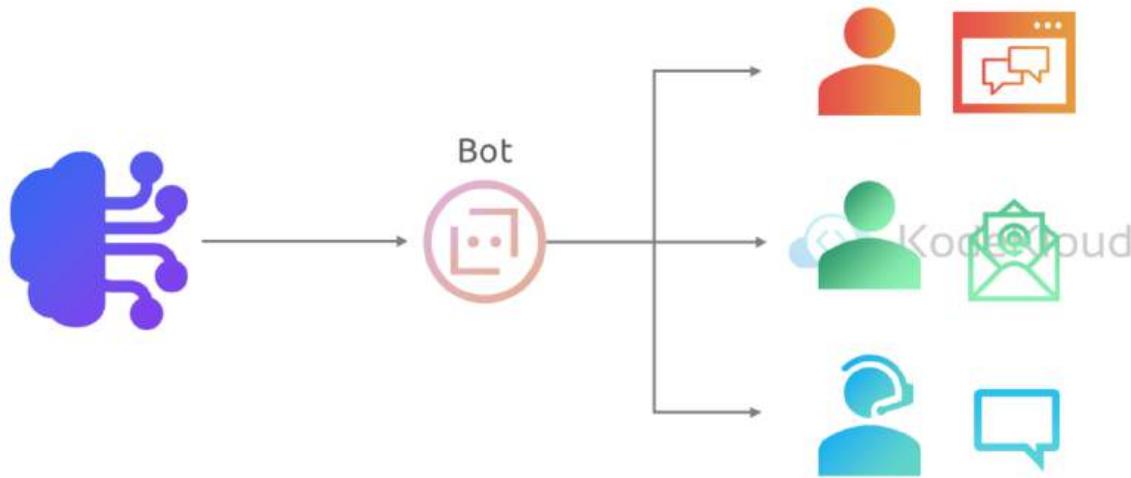
Azure Bot Service



© Copyright KodeKloud

Azure Bot Service

Azure Bot Service



© Copyright KodeKloud

Cloud-Based Platform for Developing and Managing Bots: Azure Bot Service provides a comprehensive cloud environment for creating, deploying, and managing intelligent bots that can interact with users naturally. Azure Bot Service offers a robust platform for developing sophisticated bots that can understand and respond to user inputs.

Cloud-Based Platform for Developing and Managing Bots

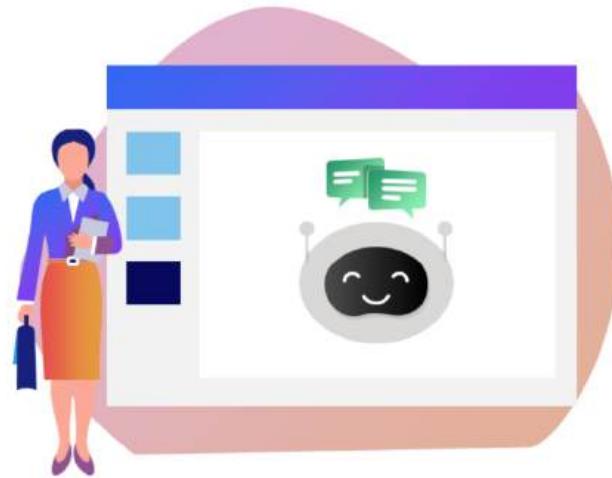


Azure Bot Service enables creating and managing intelligent bots.

© Copyright KodeKloud

Cloud-Based Platform for Developing and Managing Bots: Azure Bot Service provides a comprehensive cloud environment for creating, deploying, and managing intelligent bots that can interact with users naturally.

Cloud-Based Platform for Developing and Managing Bots



Azure Bot Service offers a platform for developing smart bots.

© Copyright KodeKloud

Azure Bot Service offers a robust platform for developing sophisticated bots that can understand and respond to user inputs.

Integration With AI Language and Other Services

Azure bots integrate with AI Language services



Natural Language
Understanding



Sentiment
Analysis

© Copyright KodeKloud

Integration with AI Language and Other Services: Bots developed using Azure can seamlessly integrate with AI Language services, enabling natural language understanding, sentiment analysis, and more. This integration enhances the bot's ability to comprehend and respond to user queries effectively.

Integration With AI Language and Other Services



Answering
questions



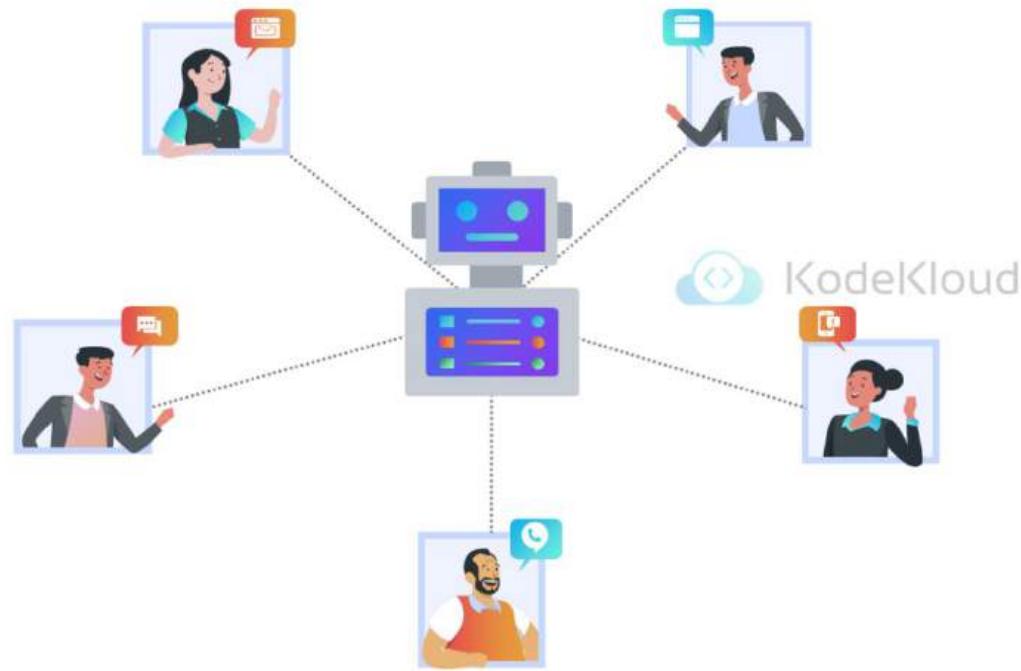
Providing
recommendations



Automating
workflows

By integrating with AI Language and other AI services, these bots can perform tasks such as answering questions, providing recommendations, and automating workflows.

Connectivity Through Multiple Channels



© Copyright KodeKloud

Connectivity Through Multiple Channels : Azure Bot Service allows bots to connect with users across various channels, including web, email, social media, and messaging platforms. This ensures that users can interact with the bots through their preferred communication channels.

Connectivity Through Multiple Channels



Broad User Reach



KodeKloud
Engagement



Enhancing customer service



Automating interactions

© Copyright KodeKloud

The service supports connectivity through multiple channels, ensuring broad user reach and engagement. This makes Azure Bot Service an essential tool for enhancing customer service and automating interactions.

Conversational Language and Understanding



© Copyright KodeKloud

Conversational Language and Understanding

Conversational Language and Understanding



© Copyright KodeKloud

Utterance: An utterance is a spoken or written command that a user gives to the system. In this example, the utterance is "Set a timer for 10 minutes." It represents the complete input provided by the user.

Conversational Language and Understanding



It is a specific detail in the utterance that provides context.



Set a timer for 10 minutes

Entity: An entity is a specific piece of information within the utterance that provides context and details about what the user is referring to. In this example, "10 minutes" is the entity, specifying the duration for the timer.

Conversational Language and Understanding

03

Intent



KodeKloud

It is the action that the user wants the system to perform.

Set a timer for 10 minutes

© Copyright KodeKloud

Intent: An intent is the action that the user wants the system to perform based on the utterance. In this example, the intent is "Set timer," indicating that the user wants the system to start a timer.

Speech Recognition and Synthesis



© Copyright KodeKloud

Speech Recognition and Synthesis

Speech Recognition and Synthesis

01

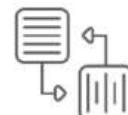
Speech Recognition

Converting speech to text

02

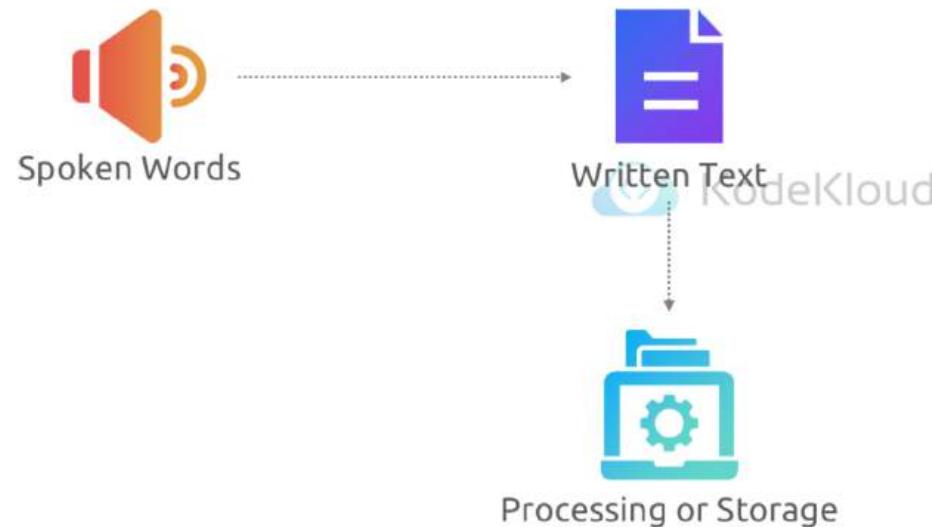
Speech Synthesis

Converting text to speech



Azure's Speech service offers robust capabilities for both speech recognition (converting speech to text) and speech synthesis (converting text to speech).

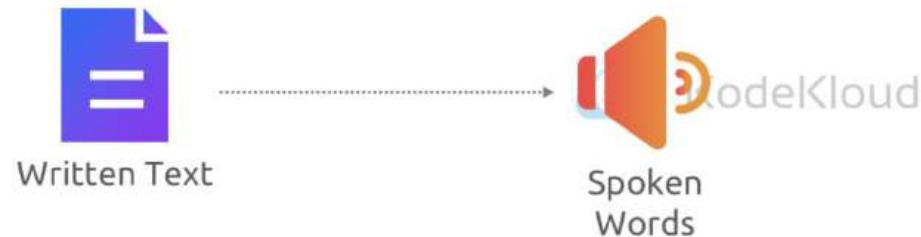
Speech Recognition



© Copyright KodeKloud

Use the speech-to-text capabilities of the Speech service to transcribe spoken words into written text. This allows users to convert audio input, such as dictation or voice commands, into text format for further processing or storage.

Speech Synthesis



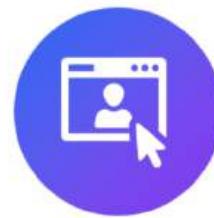
© Copyright KodeKloud

Use the text-to-speech capabilities of the Speech service to generate audible speech from written text.

Speech Synthesis



Accessibility



User Interaction

 KodeKloud

© Copyright KodeKloud

This enables applications to provide spoken feedback or read out text content, enhancing accessibility and user interaction.

Speech Recognition and Synthesis



Virtual
Assistants



Transcription
Services



Ko



Accessibility Tools

These functionalities are essential for developing applications that require voice interactions, such as virtual assistants, transcription services, and accessibility tools. By leveraging these capabilities, developers can create more interactive and user-friendly applications.

Azure AI Document Intelligence

Module Introduction

- 01 AI Document Intelligence Services
- 02 Form Analysis Using Document Intelligence
- 03 Document Intelligence Studio
- 04 Data Extraction Using Document Intelligence Studio



AI Document Intelligence Services



AI Document Intelligence Services



These services offer **powerful tools** for analyzing and extracting data from documents.



The Document Analysis service **provides structured data representations** and identifies regions of interest within documents.



Pre-built models cater to **common document types** such as invoices, receipts, and ID documents, offering out-of-the-box functionality.



Custom models can be trained using sample data to **meet specific organizational needs**, ensuring accurate and relevant data extraction.



KodeKloud

These services streamline document processing and enhance efficiency in handling large volumes of documents.

© Copyright KodeKloud

Azure's AI Document Intelligence Services offer powerful tools for analyzing and extracting data from documents. The Document Analysis service provides structured data representations and identifies regions of interest within documents. Prebuilt models cater to common document types such as invoices, receipts, and ID documents, offering out-of-the-box functionality. Custom models can be trained using sample data to meet specific organizational needs, ensuring accurate and relevant data extraction. These services streamline document processing and enhance efficiency in handling large volumes of documents.

AI Document Intelligence Services

The screenshot shows a user interface for AI Document Intelligence Services. At the top, there are buttons for "Run analysis" and "Analyze options". The main area displays the content of a document with several callout boxes highlighting specific sections. One box discusses the early stages of AI in healthcare, mentioning pharmaceutical and life sciences investments. Another box talks about enhancing patient and provider experiences. A third box discusses AI in drug discovery research and development. A fourth box highlights AI's role in automating and streamlining processes. A fifth box mentions AI's impact on supply chain management. A sixth box discusses AI's potential to reduce costs in drug discovery research and development. A seventh box emphasizes the importance of AI in providing timely information to healthcare providers. A eighth box highlights AI's ability to analyze and extract insights from structured data representations.

DOCUMENT ANALYSIS

- Structured Data Representations:** Return data in structured formats, making it easier to analyze and extract insights
- Regions of Interest and Relationships:** Identify specific areas within documents and understand their relationships, such as tables, headers, and footers
- Analyze Options:** Offers configurable options for free and chargeable analysis, allowing tailored processing based on document type and complexity



AI Document Intelligence Services

The screenshot shows a user interface for document analysis. On the left, there is a preview of a receipt from 'Contoso' with various items listed. On the right, a detailed analysis pane is open, showing the extracted fields and their confidence levels:

Field	Value	Confidence (%)
DocType	receiptDetailItem	-
Items	(2)	98.70%
MerchantName	Contoso	98.70%
Total	\$2,516.28	98.70%
MerchantAddress	123 Main Street Redmond, WA 98052	99.00%
HouseNumber	123	-
Road	Main Street	-
PostalCode	98052	-
City	Redmond	-
State	WA	-
StreetAddress	123 Main Street	-
Subtotal	\$2,297.97	99.00%
Tax	\$218.31	-
Total	\$2,516.28	-
Subtotal	\$2,297.97	99.00%



PRE-BUILT MODELS

- **Invoices:** Automatically extract relevant information from invoices, such as amounts, dates, and vendor details
- **Receipts:** Capture and process key details from receipts, including total amounts and items purchased
- **ID Documents:** Recognize and extract key-value pairs from identification documents such as passports and driver's licenses

AI Document Intelligence Services

Custom models

Extract information from forms and documents with custom models. Train a model by labeling as few as 5 example documents. (The :)

Custom form models

Custom form models work well when the target documents share a common visual layout. Training only takes a few minutes, and more than 100 languages are supported.

From: **IRS - Internal Revenue Service**
Request for Taxpayer Identification Number and Certification
Go to [IRS.gov/Form990](#) for instructions and the latest information.
1. Name (as shown on your income tax return). Name is required on this line do not leave this line blank.
Jamie Doe
2. Checkmark indicates the tax classification of the person whose name is entered on line 1. (Check only one of the following boxes.)
• Sole proprietorship or
• Corporation
• Partnership
• Individual
• Limited liability company. Enter the tax classification (LLC) if corporation, partnership, or individual.
Note: Check the appropriate box for the tax classification of the single-member owner. Do not check LLC if the LLC is classified as a single-member LLC that is disregarded from the owner unless the name of the LLC is used on the document. If the LLC is used on the document, check the LLC classification. If the LLC is disregarded from the owner, check the appropriate box for the tax classification of the owner.
• Other (see instructions)
3. Business name, street, city, state, zip code (if applicable)
Jamie Doe
4. Your address (name and address updating)
Jamie Doe
5. I am a sole proprietorship.
• My name
• My business name
6. I am a limited liability company.
• My name
• My business name
Give Form to the
requester. Do not
send to the IRS.

My Projects

[+ Create a project](#) [Delete](#)

© Copyright KodeKloud



CUSTOM MODELS

- Training with Sample Data:** Allows training of custom models using at least five sample documents, enabling the extraction of specific fields relevant to your organization
- Field Identification:** Identifies and extracts fields of interest tailored to organizational needs, enhancing data extraction accuracy and relevance

Form Analysis Using Document Intelligence



Form Analysis Using Document Intelligence



© Copyright KodeKloud

The Document Intelligence service in Azure excels at extracting and understanding information from various types of forms. By using pre-trained models for standard document types or training custom models for unique forms, the service can accurately capture and process data. Its ability to perform semantic recognition ensures that the extracted information is not just text but is understood in context, making it highly useful for automated processing and analysis.

Form Analysis Using Document Intelligence

Extract information from scanned forms in image or PDF format



Pre-Trained Models

- For common document types such as
 - Invoices
 - Receipts
 - ID cards and more



Train Custom Models

- Create custom models tailored to specific forms and documents
- Help enhance accuracy and relevance

Northwind Traders

123 Main Street

555-123-4567

2/17/2020 13:07

1	Apple	\$0.90
2	Orange	\$1.60

	Sub-Total	\$2.50
	Tax	\$0.25
	Total	\$2.75

© Copyright KodeKloud

Extract information from scanned forms in image or PDF format:

Use Pre-Trained Models: Leverage pre-trained models for common document types such as invoices, receipts, ID cards, and more.

Train Custom Models: Create custom models tailored to your specific forms and documents, enhancing accuracy and relevance.

Form Analysis Using Document Intelligence

Semantic Recognition of Form Fields

Beyond Text Extraction, models **perform semantic recognition** of form fields, understanding the context and meaning of the data, not just extracting text.



Perform semantic recognition



Understands the context and meaning

Northwind Traders

123 Main Street

555-123-4567

2/17/2020 13:07

1	Apple	\$0.90
2	Orange	\$1.60
<hr/>		
	Sub-Total	\$2.50
	Tax	\$0.25
	Total	\$2.75

Semantic Recognition of Form Fields: Beyond Text Extraction, models perform semantic recognition of form fields, understanding the context and meaning of the data, not just extracting text.

Document Intelligence Studio



Document Intelligence Studio



© Copyright KodeKloud

Document Intelligence Studio in Azure provides a user-friendly, no-code environment to explore and utilize document intelligence features. By creating and enabling a resource, users can access powerful tools for document analysis and processing. The Getting Started page guides users through the initial steps, allowing them to quickly test and understand the capabilities of different models using their own documents or provided samples.

Document Intelligence Studio

The screenshot shows the 'Get started with Document Intelligence Studio' page. At the top, there's a header bar with the text 'Azure AI | Document intelligence Studio' and 'Form Recognizer Studio'. Below the header, the main title 'Get started with Document Intelligence Studio' is displayed, followed by a brief description: 'Extract text, key-value pairs, tables, and structures from forms and documents using common layouts and prebuilt models, or create your own custom models.' A 'Learn more' link is also present. The page is divided into two main sections: 'Document analysis' and 'Prebuilt models'.
Document analysis: This section is highlighted with a dashed red border. It contains three cards: 'Read', 'Layout', and 'General documents'. Each card has an icon, a title, a brief description, and a 'Try it out' button.

- Read:** Extract printed and handwritten text along with barcodes, formulas and font styles from images and documents.
- Layout:** Extract tables, check boxes, and text from forms and documents.
- General documents:** Extract key value pairs and structure like tables and selection marks from any form or document.

Prebuilt models: This section contains four cards representing unique document types:

- Invoice
- Bill of Lading
- Bank Statement
- Passport

Each card shows an icon of the document type with a dollar sign (\$) symbol, indicating they use prebuilt models for extraction.

© Copyright KodeKloud

Document Intelligence Studio in Azure provides a user-friendly, no-code environment to explore and utilize document intelligence features. By creating and enabling a resource, users can access powerful tools for document analysis and processing. The Getting Started page guides users through the initial steps, allowing them to quickly test and understand the capabilities of different models using their own documents or provided samples.

Document Intelligence Studio

The screenshot shows the 'Get started with Document Intelligence Studio' page. At the top, there's a header bar with the text 'Azure AI | Document intelligence Studio' and a 'Form Recognizer Studio' link. Below the header, the main title 'Get started with Document Intelligence Studio' is displayed, followed by a brief description: 'Extract text, key-value pairs, tables, and structures from forms and documents using common layouts and prebuilt models, or create your own custom models.' A 'Learn more' link is also present. The page is divided into several sections:

- Document analysis**: A section describing how to extract text, tables, structure, key-value pairs, and named entities from documents.
- Read**: An icon showing a document being processed by a hand, with the description: 'Extract printed and handwritten text along with barcodes, formulas and font styles from images and documents.' A 'Try it out' button is available.
- Layout**: An icon showing a document being processed by a hand holding a ruler, with the description: 'Extract tables, check boxes, and text from forms and documents.' A 'Try it out' button is available.
- General documents**: An icon showing a document being processed by a hand holding a magnifying glass, with the description: 'Extract key value pairs and structure like tables and selection marks from any form or document.' A 'Try it out' button is available.
- Prebuilt models**: A section titled 'Extract data from unique document types using the following prebuilt models.' It contains four icons representing different document types: a green document with a barcode, a blue document with a table, a pink document with a list, and a yellow document with a signature.

© Copyright KodeKloud

Document Intelligence Studio in Azure provides a user-friendly, no-code environment to explore and utilize document intelligence features. By creating and enabling a resource, users can access powerful tools for document analysis and processing. The Getting Started page guides users through the initial steps, allowing them to quickly test and understand the capabilities of different models using their own documents or provided samples.

Document Intelligence Studio

No-Code Approach

Explore the functionality of Document Intelligence using a no-code approach. Use sample documents or your own to understand how the service operates.

Azure AI | Document Intelligence Studio

Form Recognizer Studio

Get started with Document Intelligence Studio

Extract text, key-value pairs, tables, and structures from forms and documents using common layouts and prebuilt models, or create your own custom models. [Learn more](#)

Document analysis

Extract text, tables, structure, key-value pairs, and named entities from documents.



Read
Extract printed and handwritten text along with barcodes, formulas and font styles from images and documents.
[Try it out](#)



Layout
Extract tables, check boxes, and text from forms and documents.
[Try it out](#)



General documents
Extract key value pairs and structure like tables and selection marks from any form or document.
[Try it out](#)

Prebuilt models

Extract data from unique document types using the following prebuilt models.



© Copyright KodeKloud

No Code Approach: Explore the functionality of Document Intelligence using a no-code approach. Utilize samples and your own documents to see how the service works.

Document Intelligence Studio

1 Create a Resource

Document Intelligence Resource

Set up a Document Intelligence resource to get started



AI Services Resource

Alternatively, use AI services resource to access a broader range of capabilities



Enable the Resource

Once the resource is created, enable it in Document Intelligence Studio to begin using its features

2 Getting Started Page

Navigate to the Getting Started page and select a model to try out the service's capabilities.

The screenshot shows the 'Azure AI | Document Intelligence Studio' interface. At the top, there's a navigation bar with 'Form Recognizer Studio' and a 'Get started with Document Intelligence Studio' button. Below that is a 'Document analysis' section with a sub-section for 'Read'. The 'Read' section includes three cards: 'Read' (extract text and handwritten text along with names, numbers and dates from images and documents), 'Layout' (extract tables, check boxes, and text from forms and documents), and 'General documents' (extract key-value pairs and structure like tables and sentences easily from any type of document). Further down, there's a 'Prebuilt models' section with four cards: 'Text' (extract data from unique document types using the following prebuilt models), 'Table' (extract tables from various document types), 'Form' (extract forms from various document types), and 'Image' (extract images from various document types).

© Copyright KodeKloud

First - Create a Resource

Document Intelligence Resource: Set up a Document Intelligence resource to get started.

AI Services Resource: Alternatively, use an AI services resource to access a broader range of capabilities.

Enable the Resource: Once the resource is created, enable it in Document Intelligence Studio to begin using its features.

Getting Started Page: Navigate to the Getting Started page and select a model to try out the service's capabilities.

Azure AI Search

Module Introduction

01 Introduction to Knowledge Mining

02 Azure AI Search Services

03 AI Enrichment

04 Exploring Azure AI Search Index Using UI



KodeKloud

Document Intelligence Studio



Knowledge Mining – Introduction

Organizations have a lot of content



Data is Locked Away

Information is often hidden in documents, PDFs, handwritten notes, and other unstructured formats, making it difficult to access



Time-Consuming and Labor-Intensive

Extracting and finding valuable data from sources can be extremely time-consuming and requires significant manual effort



Insights at Scale

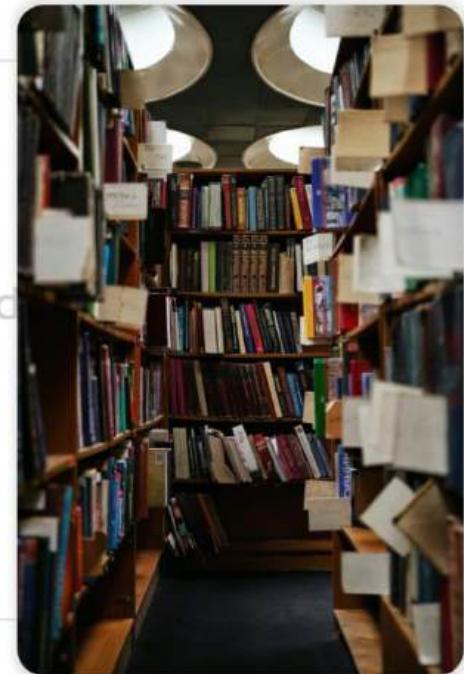
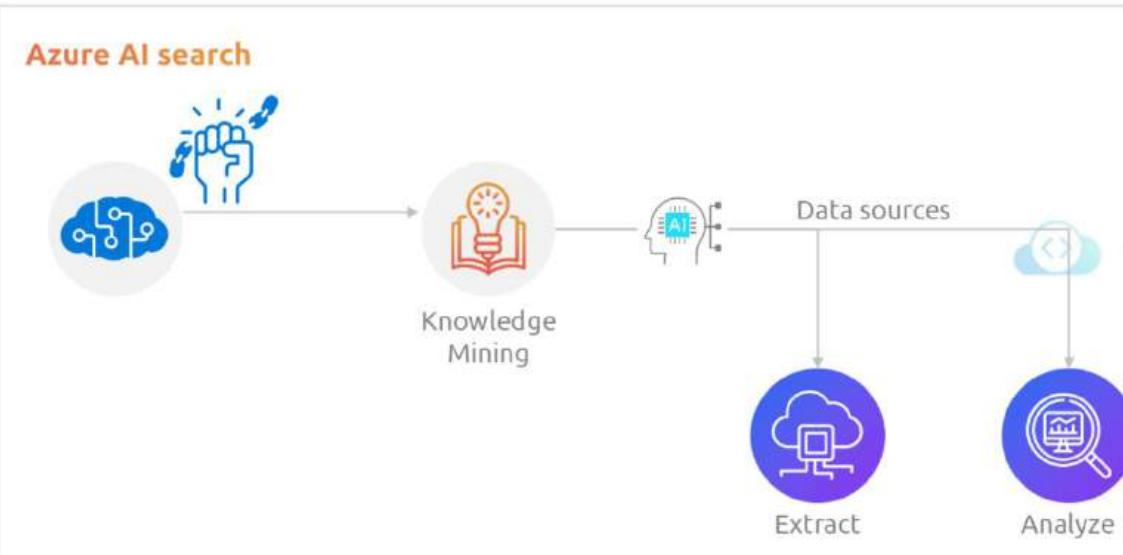
Knowledge mining utilizes AI to efficiently extract insights from vast amounts of data, achieving this on a large scale



© Copyright KodeKloud

Document Intelligence Studio in Azure provides a user-friendly, no-code environment to explore and utilize document intelligence features. By creating and enabling a resource, users can access powerful tools for document analysis and processing. The Getting Started page guides users through the initial steps, allowing them to quickly test and understand the capabilities of different models using their own documents or provided samples.

Knowledge Mining – Introduction



© Copyright KodeKloud

Azure AI Search: Azure AI Search is a powerful platform designed for knowledge mining, providing AI-driven capabilities to extract and analyze information from various data sources.

Knowledge Mining – Introduction



© Copyright KodeKloud

Solutions Provide Access Through:

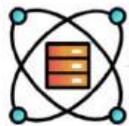
Search and Bots: Utilize search functionalities and AI bots to answer questions and retrieve information quickly.

Dashboards: Create interactive dashboards that display insights and key metrics derived from mined data.

Knowledge Mining – Introduction

Solutions provide access through:

Business Applications



Decision
making



Operational
efficiency

Further Analysis



Analysis



Insights

Data



Analytics &
reporting



© Copyright KodeKloud

Solutions Provide Access Through:

Business Applications: Integrate knowledge mining capabilities into business applications to enhance decision-making and operational efficiency.

Further Analysis: Make data available for deeper analysis and insights, supporting advanced analytics and reporting.

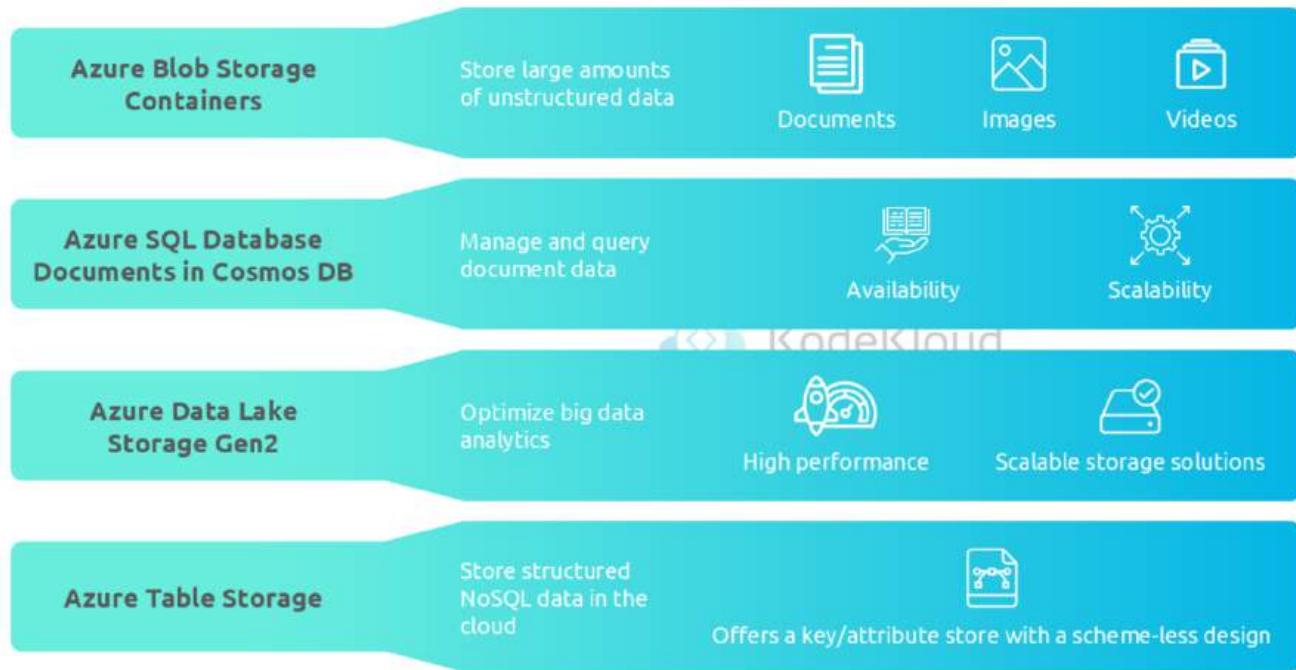
Azure AI Search Services



Azure AI Search Services



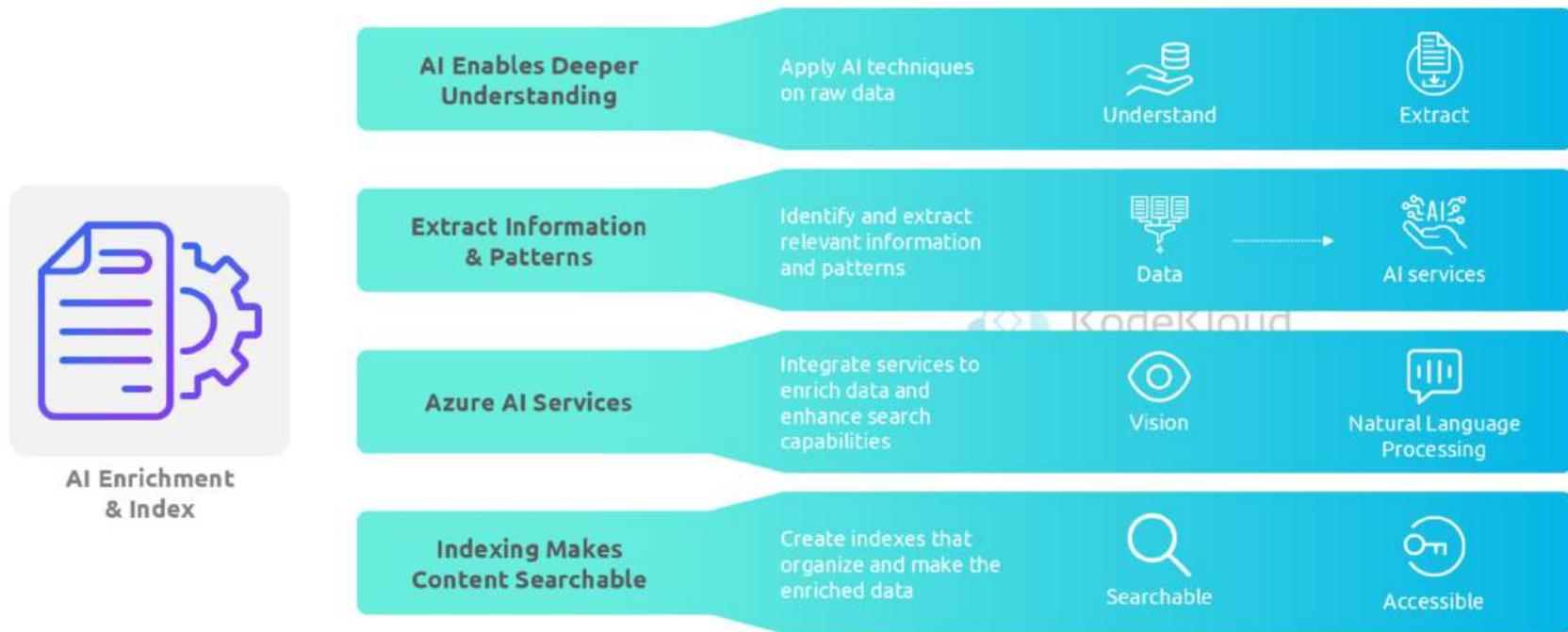
Ingest



© Copyright KodeKloud

- Azure Blob Storage Containers: Store large amounts of unstructured data, such as documents, images, and videos.
- Azure SQL Database Documents in Cosmos DB: Manage and query document data with high availability and scalability.
- Azure Data Lake Storage Gen2: Optimize big data analytics by providing high-performance, secure, and scalable storage solutions.
- Azure Table Storage: Store structured NoSQL data in the cloud, offering a key/attribute store with a schema-less design.

Azure AI Search Services



© Copyright KodeKloud

AI Enables Deeper Understanding: Apply AI techniques to understand and extract meaningful information from raw data.

Extract Information & Patterns: Identify and extract relevant information and patterns from data using AI services.

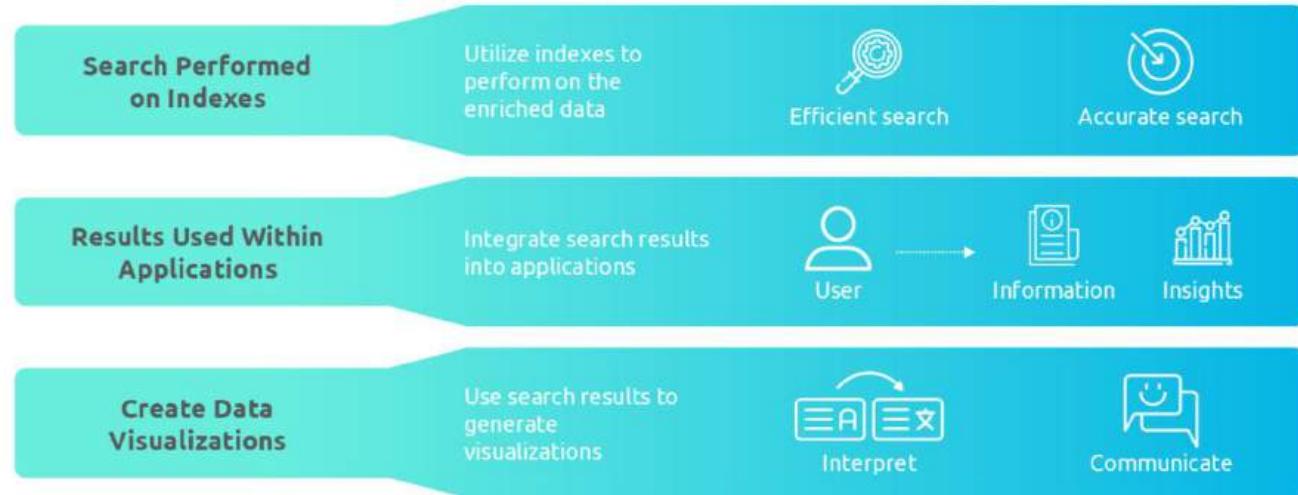
Azure AI Services: Integrate services like Vision, Natural Language Processing, and others to enrich data and enhance search capabilities.

Indexing Makes Content Searchable: Create indexes that organize and make the enriched data easily searchable and accessible.

Azure AI Search Services



Explore



© Copyright KodeKloud

Search Performed on Indexes: Utilize indexes to perform efficient and accurate searches on the enriched data.

Results Used Within Applications: Integrate search results into applications to provide users with relevant information and insights.

Create Data Visualizations: Use search results to generate visualizations, helping to interpret and communicate insights effectively.

AI Enrichment

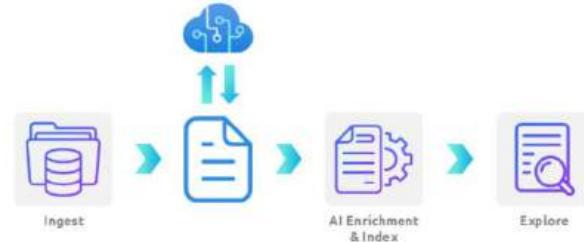


AI Enrichment



AI Enrichment

In Azure AI Search



Enhancing Content for Search Purposes



AI Enrichment



Content

- More useful
- Relevant for search



Structured Data



Insights

© Copyright KodeKloud

Enhancing Content for Search Purposes: AI enrichment processes make content more useful and relevant for search by adding structured data and insights.

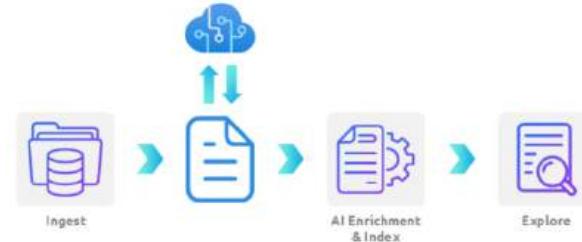
Serialized Data for Indexing: The processed and enriched data is serialized and passed to the search engine for efficient indexing.

Knowledge Store

Data Sink Created by AI Search Enrichment Pipeline: Acts as a repository for enriched data, created and maintained by the AI Search enrichment pipeline, enabling advanced search and analytics capabilities.

AI Enrichment

In Azure AI Search



Creating Enriched Content with Skillsets



Recognize Entities in Text

Identifies and extracts key entities such as names, dates, and locations from the text



Translate Text

Converts text from one language to another, broadening accessibility and understanding



Evaluate Sentiment

Analyzes the sentiment expressed in the text, determining whether it is positive, negative, or neutral

Producing Enriched Documents - A skillset processes raw data to generate enriched documents



Consumed During Indexing

Enriched documents are utilized in the indexing process to improve search accuracy and relevance



Projected to a Knowledge Store

Enriched data is stored in a knowledge store, making it accessible for further analysis and use

© Copyright KodeKloud

Creating Enriched Content with Skillsets

Recognize Entities in Text: Identifies and extracts key entities such as names, dates, and locations from the text.

Translate Text: Converts text from one language to another, broadening accessibility and understanding.

Evaluate Sentiment: Analyzes the sentiment expressed in the text, determining whether it is positive, negative, or neutral.

Producing Enriched Documents

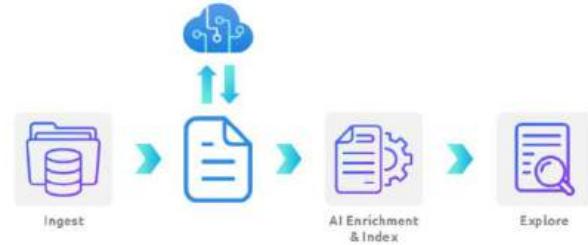
A skillset processes raw data to generate enriched documents:

Consumed During Indexing: Enriched documents are utilized in the indexing process to improve search accuracy and relevance.

Projected to a Knowledge Store: Enriched data is stored in a knowledge store, making it accessible for further analysis and use.

AI Enrichment

In Azure AI Search



Serialized Data for Indexing

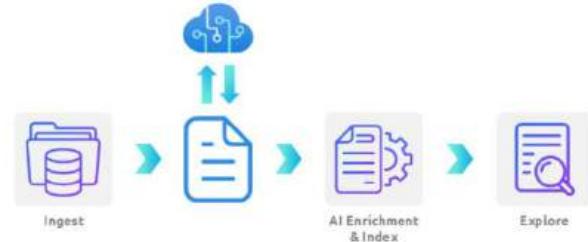


© Copyright KodeKloud

Serialized Data for Indexing: The processed and enriched data is serialized and passed to the search engine for efficient indexing.

AI Enrichment

In Azure AI Search

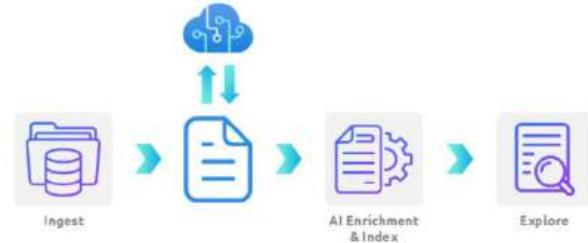


© Copyright KodeKloud

Serialized Data for Indexing: The processed and enriched data is serialized and passed to the search engine for efficient indexing.

AI Enrichment

In Azure AI Search



Knowledge Store

Data sink created by AI Search enrichment pipeline



© Copyright KodeKloud

Knowledge Store

Data Sink Created by AI Search Enrichment Pipeline: Acts as a repository for enriched data, created and maintained by the AI Search enrichment pipeline, enabling advanced search and analytics capabilities.

Generative AI

Module Introduction

- 01 Introduction to Generative AI
- 02 Language Models
- 03 Language Model Training
- 04 Foundation Models



Module Introduction

- 05 Large and Small Language Models
- 06 Copilots
- 07 Microsoft Copilot Services
- 08 Copilot Prompts – Considerations



KodeKloud

Module Introduction

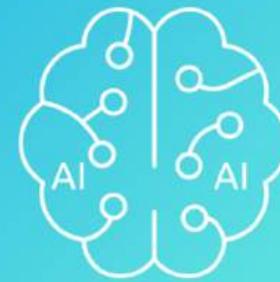
09

Developing and Extending Copilot



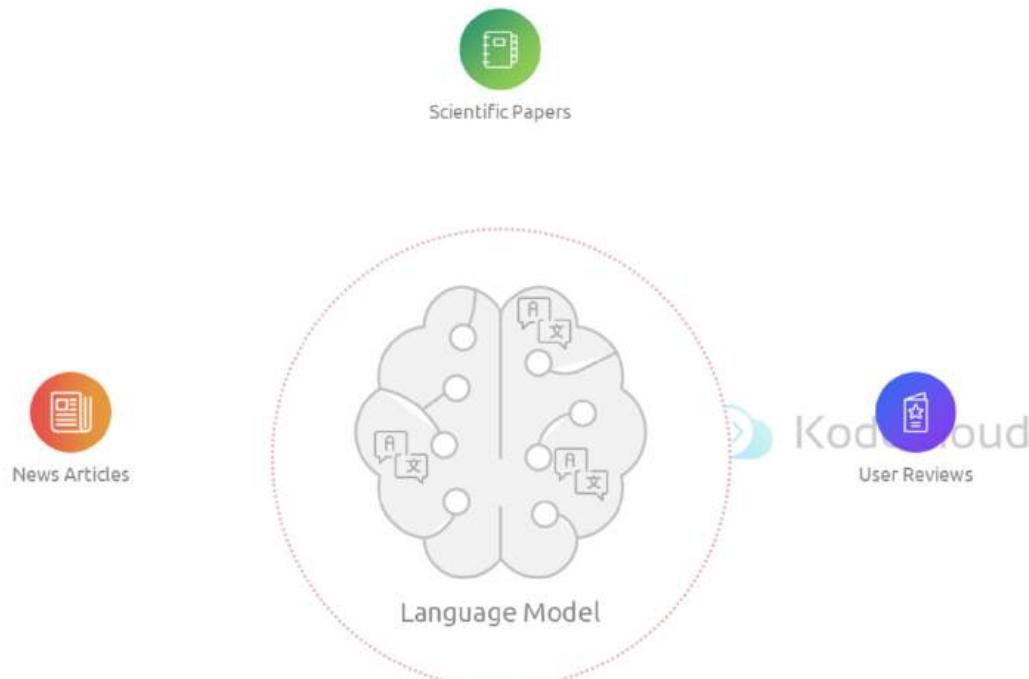
KodeKloud

Generative AI – Introduction



Language Models



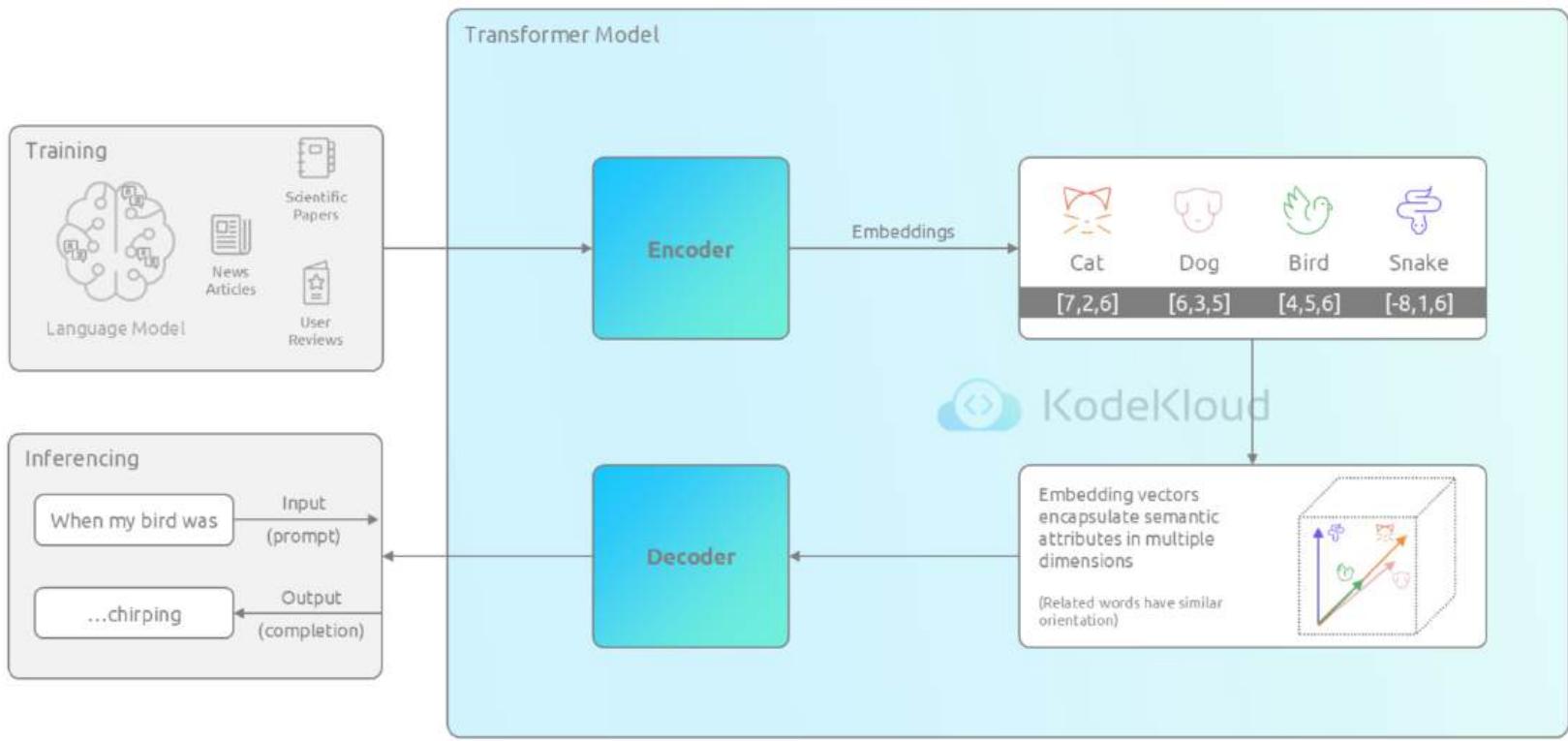


© Copyright KodeKloud

Training

Training Text

Use diverse and extensive text sources such as news articles, scientific papers, and user reviews to train the language model.



© Copyright KodeKloud

Transformer Model

Encoder

Encodes text tokens as vectors (embeddings).

Embedding vectors encapsulate semantic attributes in multiple dimensions.

Example: Similar words have embeddings with similar orientations.

Word Embedding Vectors
 sunny[7, 2, 6]
 cloudy[6, 3, 5]
 rainy[4, 5, 6]
 warm[8, 1, 7]

Decoder

Generates natural language sequences from the encoded vectors.

Inferencing

Input (Prompt)

Example: "The weather today is..."

Output (Completion)

Example: "...sunny and warm."

Explanation:

Language models, particularly those based on transformer architecture, involve two main components: the encoder and the decoder. During training, the model learns from a diverse and extensive text corpus. The encoder transforms input text into embeddings, which are multi-dimensional vectors representing the semantic meaning of words. The decoder then uses these embeddings to generate natural language sequences. This process enables the model to understand and generate human-like text, making it useful for a wide range of natural language processing tasks.

Language Model Training



Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

The first step in training a transformer model is to decompose **the training text** into **tokens**.

© Copyright KodeKloud

Training

Training Text

Use diverse and extensive text sources such as news articles, scientific papers, and user reviews to train the language model.

Transformer Model

Encoder

Encodes text tokens as vectors (embeddings).

Embedding vectors encapsulate semantic attributes in multiple dimensions.

Example: Similar words have embeddings with similar orientations.

WordEmbedding Vectors sunny[7, 2, 6] cloudy[6, 3, 5] rainy[4, 5, 6] warm[8, 1, 7]

Decoder

Generates natural language sequences from the encoded vectors.

Inferencing

Input (Prompt)

Example: "The weather today is..."

Output (Completion)

Example: "...sunny and warm."

Explanation:

Language models, particularly those based on transformer architecture, involve two main components: the encoder and the decoder. During training, the model learns from a diverse and extensive text corpus. The encoder transforms input text into embeddings, which are multi-dimensional vectors representing the semantic meaning of words. The decoder then uses these embeddings to generate natural language sequences. This process enables the model to understand and generate human-like text, making it useful for a wide range of natural language processing tasks.

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Example sentence

I heard a bird chirping in a tree

1 2 3 4 5 6 7



© Copyright KodeKloud

Training

Training Text

Use diverse and extensive text sources such as news articles, scientific papers, and user reviews to train the language model.

Transformer Model

Encoder

Encodes text tokens as vectors (embeddings).

Embedding vectors encapsulate semantic attributes in multiple dimensions.

Example: Similar words have embeddings with similar orientations.

WordEmbedding Vectors sunny[7, 2, 6] cloudy[6, 3, 5] rainy[4, 5, 6] warm[8, 1, 7]

Decoder

Generates natural language sequences from the encoded vectors.

Inferencing

Input (Prompt)

Example: "The weather today is..."

Output (Completion)

Example: "...sunny and warm."

Explanation:

Language models, particularly those based on transformer architecture, involve two main components: the encoder and the decoder. During training, the model learns from a diverse and extensive text corpus. The encoder transforms input text into embeddings, which are multi-dimensional vectors representing the semantic meaning of words. The decoder then uses these embeddings to generate natural language sequences. This process enables the model to understand and generate human-like text, making it useful for a wide range of natural language processing tasks.

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Example sentence

I heard a dog
1 2 3 4



© Copyright KodeKloud

Training

Training Text

Use diverse and extensive text sources such as news articles, scientific papers, and user reviews to train the language model.

Transformer Model

Encoder

Encodes text tokens as vectors (embeddings).

Embedding vectors encapsulate semantic attributes in multiple dimensions.

Example: Similar words have embeddings with similar orientations.

WordEmbedding Vectors sunny[7, 2, 6] cloudy[6, 3, 5] rainy[4, 5, 6] warm[8, 1, 7]

Decoder

Generates natural language sequences from the encoded vectors.

Inferencing

Input (Prompt)

Example: "The weather today is..."

Output (Completion)

Example: "...sunny and warm."

Explanation:

Language models, particularly those based on transformer architecture, involve two main components: the encoder and the decoder. During training, the model learns from a diverse and extensive text corpus. The encoder transforms input text into embeddings, which are multi-dimensional vectors representing the semantic meaning of words. The decoder then uses these embeddings to generate natural language sequences. This process enables the model to understand and generate human-like text, making it useful for a wide range of natural language processing tasks.

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

The semantic relationship between **tokens** is encoded in **vectors**, known as **embeddings**.

© Copyright KodeKloud

While it may be convenient to represent tokens as simple IDs - essentially creating an index for all the words in the vocabulary, they don't tell us anything about the meaning of the words, or the relationships between them. To create a vocabulary that encapsulates semantic relationships between the tokens, we define contextual vectors, known as embeddings, for them. Vectors are multi-valued numeric representations of information, for example [10, 3, 1] in which each numeric element represents a particular attribute of the information. For language tokens, each element of a token's vector represents some semantic attribute of the token. The specific categories for the elements of the vectors in a

language model are determined during training based on how commonly words are used together or in similar contexts.

Vectors represent lines in multidimensional space, describing direction and distance along multiple axes (you can impress your mathematician friends by calling these amplitude and magnitude). It can be useful to think of the elements in an embedding vector for a token as representing steps along a path in multidimensional space. For example, a vector with three elements represents a path in 3-dimensional space in which the element values indicate the units traveled forward/back, left/right, and up/down. Overall, the vector describes the direction and distance of the path from origin to end.

The elements of the tokens in the embeddings space each represent some semantic attribute of the token, so that semantically similar tokens should result in vectors that have a similar orientation – in other words they point in the same direction. A technique called cosine similarity is used to determine if two vectors have similar directions (regardless of distance), and therefore represent semantically linked words. For example, the embedding vectors for "dog" and "puppy" describe a path along an almost identical direction, which is also fairly similar to the direction for "cat". The embedding vector for "skateboard" however describes journey in a very different direction.

Note:

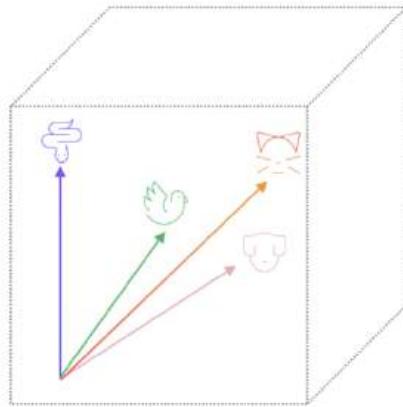
The example shows a simple model in which each embedding has only three dimensions. Real language models have many more dimensions.

There are multiple ways you can calculate appropriate embeddings for a given set of tokens, including language modeling algorithms like Word2Vec or the encoder block in a transformer model.

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention



Token	Cat	Dog	Bird	Snake
Embeddings	9	8	4	10
	[7,2,6]	[6,3,5]	[4,5,6]	[-8,1,6]

© Copyright KodeKloud

While it may be convenient to represent tokens as simple IDs - essentially creating an index for all the words in the vocabulary, they don't tell us anything about the meaning of the words, or the relationships between them. To create a vocabulary that encapsulates semantic relationships between the tokens, we define contextual vectors, known as embeddings, for them. Vectors are multi-valued numeric representations of information, for example [10, 3, 1] in which each numeric element represents a particular attribute of the information. For language tokens, each element of a token's vector represents some semantic attribute of the token. The specific categories for the elements of the vectors in a

language model are determined during training based on how commonly words are used together or in similar contexts.

Vectors represent lines in multidimensional space, describing direction and distance along multiple axes (you can impress your mathematician friends by calling these amplitude and magnitude). It can be useful to think of the elements in an embedding vector for a token as representing steps along a path in multidimensional space. For example, a vector with three elements represents a path in 3-dimensional space in which the element values indicate the units traveled forward/back, left/right, and up/down. Overall, the vector describes the direction and distance of the path from origin to end.

The elements of the tokens in the embeddings space each represent some semantic attribute of the token, so that semantically similar tokens should result in vectors that have a similar orientation – in other words they point in the same direction. A technique called cosine similarity is used to determine if two vectors have similar directions (regardless of distance), and therefore represent semantically linked words. For example, the embedding vectors for "dog" and "puppy" describe a path along an almost identical direction, which is also fairly similar to the direction for "cat". The embedding vector for "skateboard" however describes journey in a very different direction.

Note:

The example shows a simple model in which each embedding has only three dimensions. Real language models have many more dimensions.

There are multiple ways you can calculate appropriate embeddings for a given set of tokens, including language modeling algorithms like Word2Vec or the encoder block in a transformer model.

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Capture the strength of relationships between
tokens using the **attention** technique.

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Goal

Bird

Process

I heard a bird

Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping



I heard a bird
[1,5,6,2,] [2,9,3,1,] [3,1,1,2,] [4,10,3,2,]

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization**Step 02: Embeddings****Step 03: Attention**

Goal

Bird

Process

I heard a bird

Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping



I	heard	a	bird	???????
[1,5,6,2,]	[2,9,3,1,]	[3,1,1,2,]	[4,10,3,2,]	[5,2,2,2,]

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization**Step 02: Embeddings****Step 03: Attention**

Goal

Bird

Process

I heard a bird

Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping



I	heard	a	bird	???????
[1,5,6,2,]	[2,9,3,1,]	[3,1,1,2,]	[4,10,3,2,]	[5,2,2,2,]

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Goal

Bird

Process

I heard a bird

Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping



| heard a bird ????????

[1,5,6,2, **1**] [2,9,3,1, **2**] [3,1,1,2, **3**] [4,10,3,2, **4**] [5,2,3,?,]

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization**Step 02: Embeddings****Step 03: Attention**

Goal

Bird

Process

I heard a bird

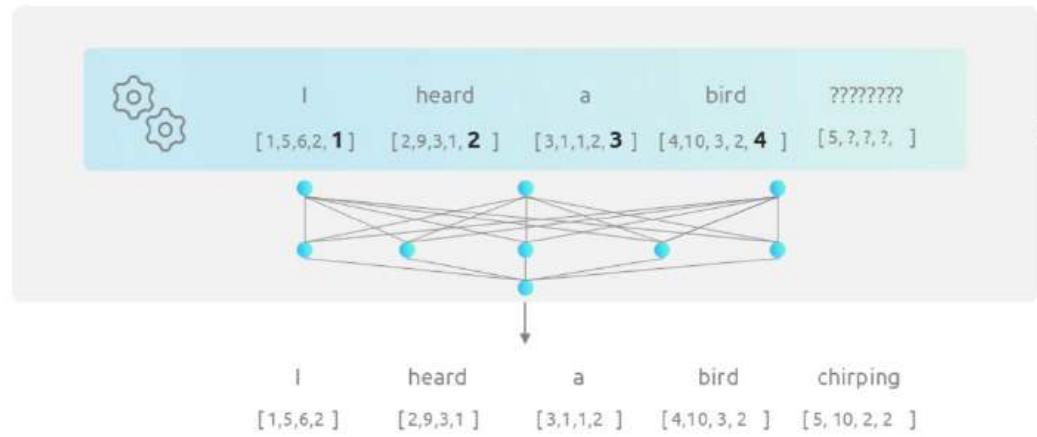
Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Step 01: Tokenization

Step 02: Embeddings

Step 03: Attention

Goal

Bird

Process

I heard a bird

Weighting

heard | bird

Possible Tokens

bird

Prediction

chirping



I	heard	a	bird	chirping
[1,5,6,2]	[2,9,3,1]	[3,1,1,2]	[4,10,3,2]	[5,10,2,2]

© Copyright KodeKloud

Capture the strength of relationships between tokens using the attention technique.

Example:

Goal: Predict the token following "bird."

Process: Represent the sentence "I noticed a bird" as vectors.

Weighting: Assign greater importance to "noticed" and "bird."

Possible Tokens: Several tokens can follow "bird."

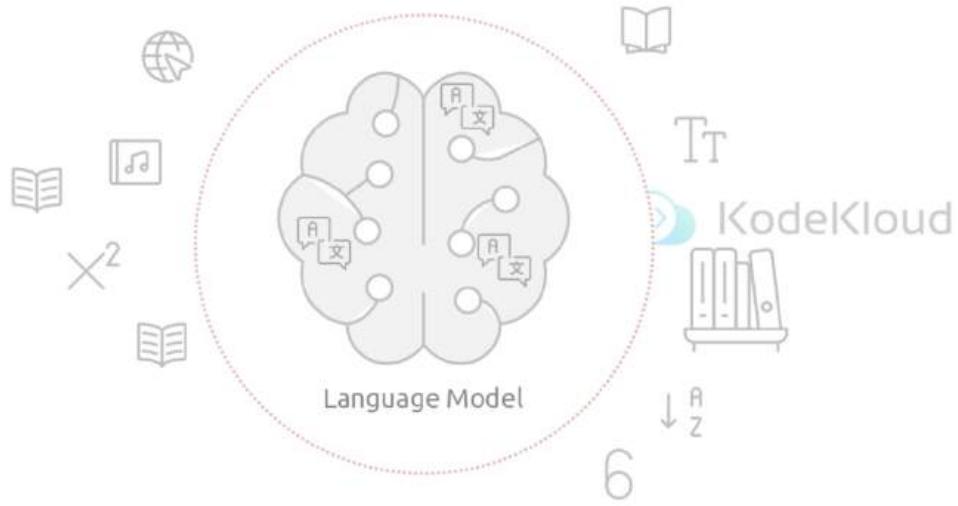
Prediction: The most likely token is added to the sequence, in this case, "singing."

Explanation:

The attention mechanism helps in predicting the next word in a sequence by focusing on the most relevant words in the context. In this example, the sentence "I noticed a bird" is encoded into vectors. The attention model prioritizes the words "noticed" and "bird" to predict the most probable next word. Among various possibilities, "singing" is identified as the most likely word to follow "bird."

Foundation Models





© Copyright KodeKloud

Organizations and developers can train their own language models from scratch, but in most cases it's more practical to use an existing foundation model, and optionally fine-tune it with your own training data. There are many sources of model that you can use.

The screenshot shows the Azure Model Catalog interface. At the top, there are three navigation tabs: 'Model Catalog' (selected), 'Azure AI Studio' (with a blue icon of two overlapping shapes), and 'Azure Machine Learning Studio' (with a blue icon of a triangle). Below these, under 'Model Catalog', is the 'Azure OpenAI' section. It features four categories: 'GPT Models', 'Image Generation Models', 'Embedding Models', and 'Speech Recognition Models'. Underneath this is the 'Open-Source Model' section, which lists several providers: Microsoft, OpenAI, HuggingFace, Mistral, Meta, Databricks, and Others... Each provider name is enclosed in a light blue rectangular box.

© Copyright KodeKloud

On Microsoft Azure, the Azure OpenAI Service includes a curated set of models from OpenAI, hosted in Azure. This offers the benefit of cutting-edge language models like the generative pre-trained transformer (GPT) collection of models (on which ChatGPT and Microsoft's own generative AI services are based) as well as the DALL-E model for image generation. Using these models from the Azure OpenAI service means that you also get the benefit of a secure, scalable Azure cloud platform in which the models are hosted.

The Azure OpenAI models are included in the Model Catalog, a curated source of models for data scientists and developers using Azure AI Studio and Azure Machine Learning. In addition to the Azure OpenAI models, the model catalog includes the latest open-source models from Microsoft and multiple partners, including:

- OpenAI
- HuggingFace
- Mistral
- Meta
- and others.

Large and Small Language Models



Large Language Models (LLMs)

Trained with **large volumes** of general text data

Contain many **billions** of parameters

Offer **comprehensive** language generation capabilities across multiple contexts

Large size can impact **performance and portability**

Time-consuming (and expensive) to fine-tune with your own training data

Example:



Small Language Models (SLMs)

Trained with **focused text** data

Contain **fewer** parameters

Provide **focused** language generation capabilities in specialized contexts

Fast and portable

Faster (and less expensive) to fine-tune with your own training data

Example:



Large Language Models (LLMs) and Small Language Models (SLMs) serve different purposes based on their size, training data, and capabilities. LLMs, with their extensive parameters and broad training data, are suitable for comprehensive language tasks but require significant resources for fine-tuning. SLMs, on the other hand, are optimized for specific tasks, with fewer parameters, making them more efficient and easier to deploy and fine-tune. Both types of models have their unique advantages and applications in the field of AI and language processing.

Exploring Copilots



Generative AI-powered assistants
integrated into applications, often as
chat interfaces

The screenshot shows the Microsoft Copilot AI interface. At the top, there's a navigation bar with icons for Home, AI, Copilot, and Help. Below it, the title "Copilot" is displayed with the subtitle "Your everyday AI companion with Bing". There are several cards for different AI tasks:

- Write**: Write the outline of a book report to help me get started.
- Compare**: I'm thinking of making a career change. Can you help me go through pros and cons?
- Code**: Write a Python script to perform Binary search.
- Create**: Write a song that celebrates the beauty of Earth.
- Chat**: What's something fun that I can do tonight?
- Travel**: Where should I travel if I have pollen allergies?
- Organize**: Create a table that helps me plan meals for the next two weeks.

At the bottom, there's a "Choose a conversation style" section with three options: "More Creative" (selected), "More Balanced", and "More Precise". A tooltip for "More Creative" says: "New chat lets you begin a fresh conversation with Bing on any subject". The main chat area has a placeholder "Ask me anything..." and a character icon. The bottom right corner shows "0/4000" and a send button icon.

Generative AI-powered assistants integrated into applications, often as chat interfaces

Functions

Contextual assistance with everyday tasks

Business users can enhance productivity and creativity

The screenshot shows the Microsoft Copilot AI interface. At the top, there's a colorful square icon followed by the word "Copilot" with a small AI icon next to it. Below that, it says "Your everyday AI companion with Bing". There are six cards arranged in two rows of three:

- Write**: Write the outline of a book report to help me get started.
- Compare**: I'm thinking of making a career change. Can you help me go through pros and cons?
- Code**: Write a Python script to perform Binary search.
- Create**: Write a song that celebrates the beauty of Earth.
- Chat**: What's something fun that I can do tonight?
- Travel**: Where should I travel if I have pollen allergies?
- Organize**: Create a table that helps me plan meals for the next two weeks.

At the bottom, there's a "Preview" button, a note about AI-powered responses, and a "Choose a conversation style" section with three options: "More Creative" (selected), "More Balanced", and "More Precise". A tooltip for "New chat" appears above the input field, which contains "Ask me anything...". The bottom right corner shows a character icon, a file icon, and some status indicators.

© Copyright KodeKloud

Functions:

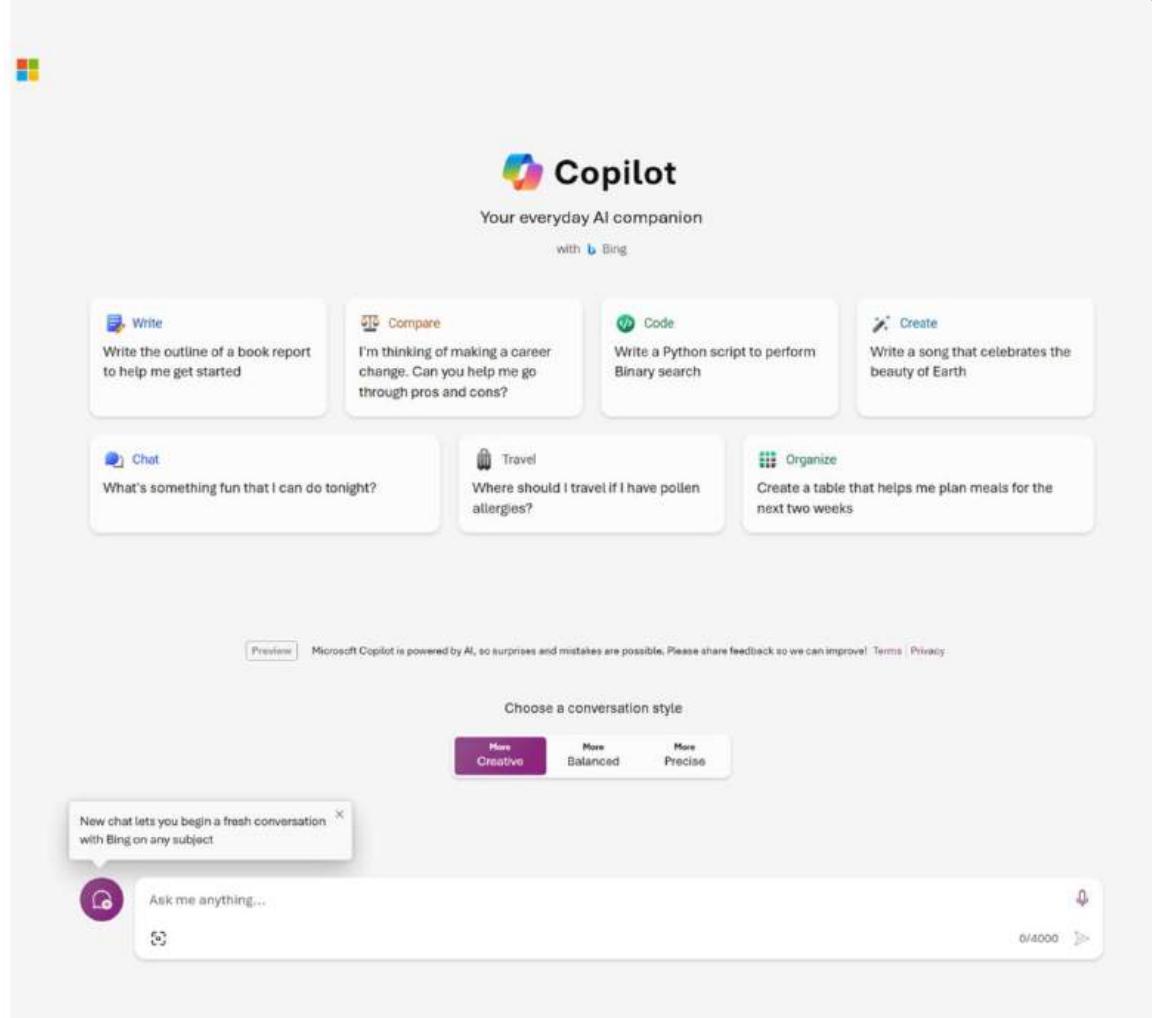
Copilots offer contextual assistance with everyday tasks, utilizing language models to interpret and generate natural language.

Business users can leverage copilots to enhance productivity and creativity through AI-generated content and task automation.

Developers

Create Plug-ins

Custom Copilots



© Copyright KodeKloud

For Developers:

Extend copilots by creating plug-ins that integrate them into business processes and data workflows.

Develop custom copilots to incorporate generative AI capabilities into apps and services.

Explanation:

Copilots are AI-driven assistants designed to provide support within applications, commonly through chat interfaces. They help interpret and generate natural language, making them valuable for a variety of tasks. Business users benefit from

increased productivity and creativity, while developers can enhance and customize these assistants to meet specific needs.

Copilot Adoption Levels

01 Enhance productivity with AI Copilot

- Utilize copilots
- Empower innovation
- Prioritize strategy

02 Extend AI Copilot with custom integration

- Custom plug-ins
- Actionable insights
- Third-party plug-ins



KodeKloud

03 Develop personalized Copilots

- Generative AI
- Engaging experiences
- Full control

Levels of Copilot Adoption

Enhance Productivity with AI Copilot

Utilize copilots to streamline daily tasks and optimize tool usage.

Empower employees to innovate and enhance their creative potential.

Focus more on strategic activities by reducing time spent on routine tasks.

Extend AI Copilot with Custom Integrations

Integrate specific business tasks and processes using custom plug-ins.

Derive actionable insights from your data and systems within productivity applications.

Develop third-party plug-ins for enhanced services and broader usability.

Develop Personalized Copilots

Embed generative AI into custom applications and workflows.

Create engaging customer experiences and bespoke commercial products.

Maintain full control over the design and development of your AI solutions.

Explanation:

Adopting AI copilots can be done at various levels, from enhancing productivity with existing tools to developing fully customized AI solutions. By integrating AI copilots, businesses can optimize operations, empower employees, and create innovative customer experiences.

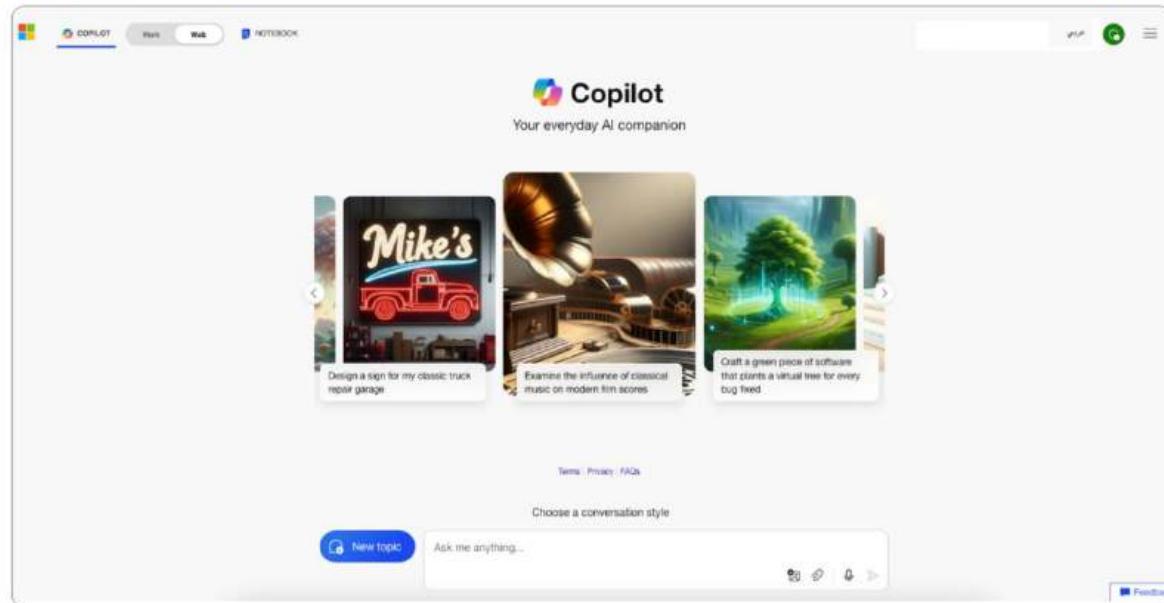
Microsoft Copilot Services



Daily Tasks

Business Processes

Infra, Security, and Software Development



AI on the web

Open to all

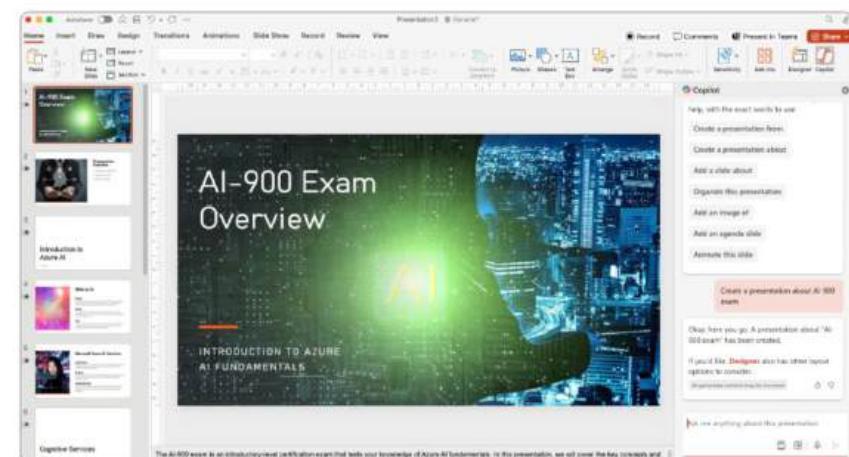
Query, create, and search

copilot.example.com

Daily Tasks

Business Processes

Infra, Security, and Software Development



AI in Microsoft 365

Subscription

Write, edit, and analyze

Create and enhance presentation

Get support

Integrate with Word, PowerPoint,
Outlook, and Teams

© Copyright KodeKloud

AI in Productivity Suites

Accessibility: Available with a subscription.

Usage:

Write, edit, and analyze documents in word processors.

Create and enhance presentations.

Get support with communication through email and messaging platforms.

Access: Integrated within popular productivity tools like Word, PowerPoint, Outlook, and Teams.

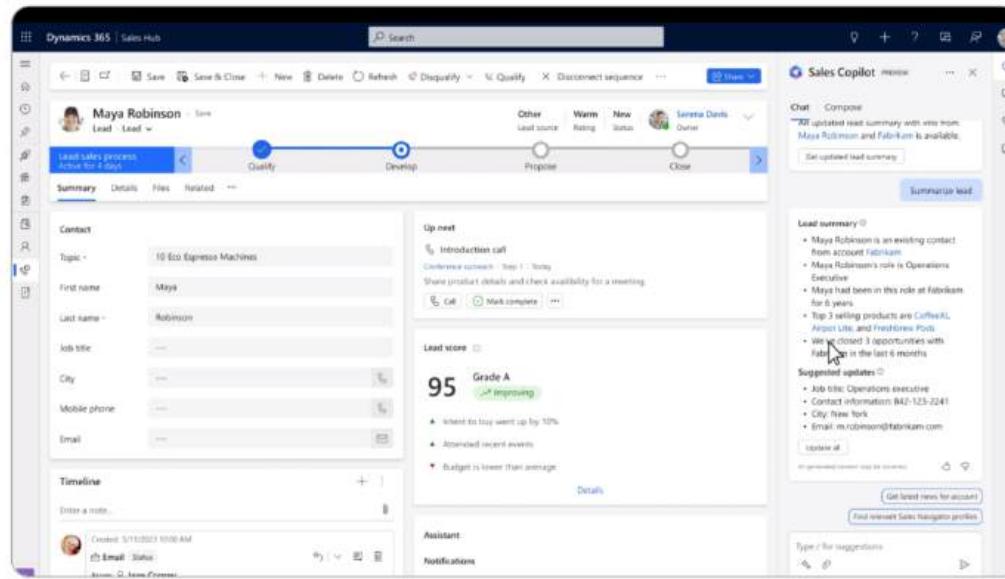
Explanation:

Copilot examples demonstrate how generative AI can be seamlessly integrated into various platforms to enhance productivity and communication. Whether on the web or within productivity suites, Copilots provide users with intelligent assistance to streamline their tasks and improve efficiency.

Daily Tasks

Business Processes

Infra, Security, and Software Development



AI in Microsoft Dynamics 365

Find relevant customer information,
quality leads, and prepare proposals

Quickly research and resolve customer
issues

© Copyright KodeKloud

AI on the Web

Accessibility: Open to all users.

Usage: Utilize Copilot to answer queries, create content, and search the internet.

Access: Visit copilot.example.com

AI in Productivity Suites

Accessibility: Available with a subscription.

Usage:

Write, edit, and analyze documents in word processors.

Create and enhance presentations.

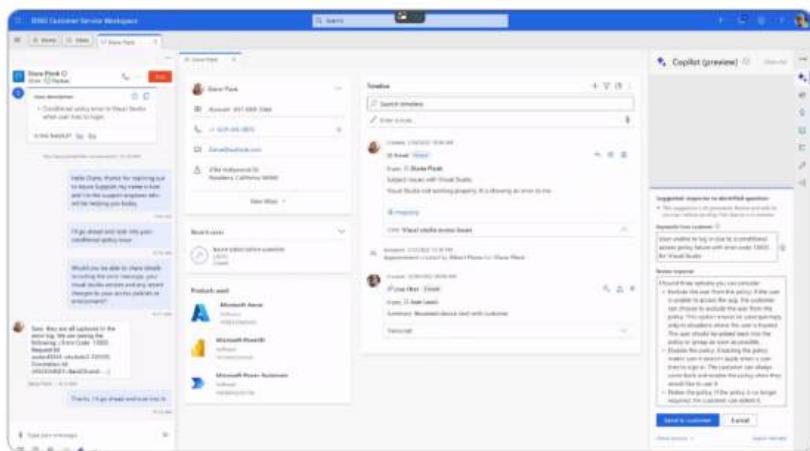
Get support with communication through email and messaging platforms.

Access: Integrated within popular productivity tools like Word, PowerPoint, Outlook, and Teams.

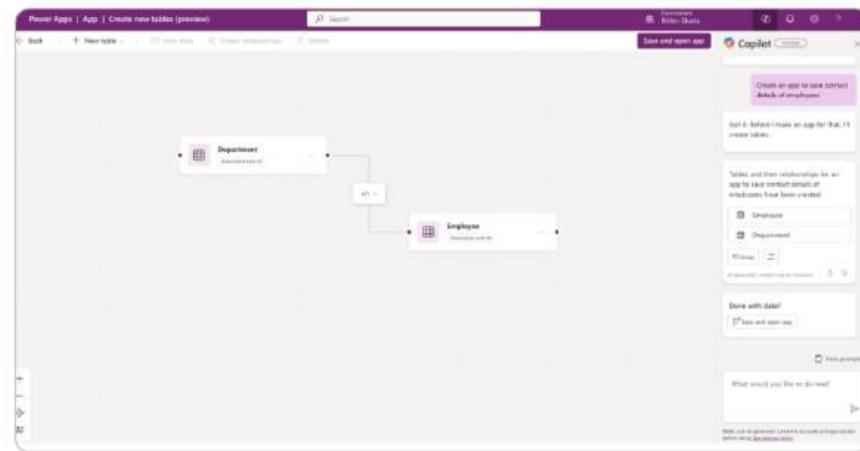
Explanation:

Copilot examples demonstrate how generative AI can be seamlessly integrated into various platforms to enhance productivity and communication. Whether on the web or within productivity suites, Copilots provide users with intelligent assistance to streamline their tasks and improve efficiency.

Daily Tasks



Business Processes



Infra, Security, and Software Development

AI in Power Platform and Microsoft Fabric

Develop Apps

Power BI to write code

© Copyright KodeKloud

AI on the Web

Accessibility: Open to all users.

Usage: Utilize Copilot to answer queries, create content, and search the internet.

Access: Visit copilot.example.com

AI in Productivity Suites

Accessibility: Available with a subscription.

Usage:

Write, edit, and analyze documents in word processors.

Create and enhance presentations.

Get support with communication through email and messaging platforms.

Access: Integrated within popular productivity tools like Word, PowerPoint, Outlook, and Teams.

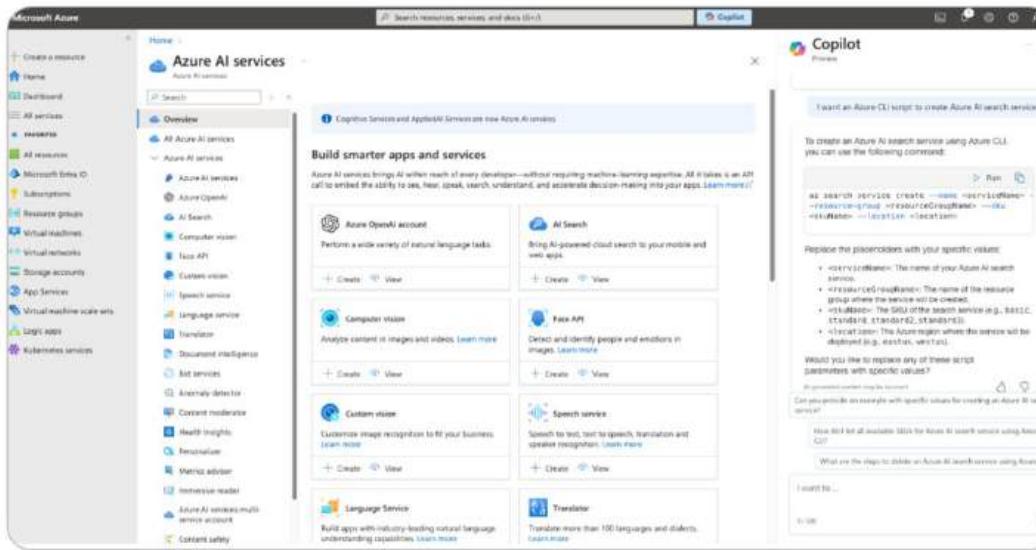
Explanation:

Copilot examples demonstrate how generative AI can be seamlessly integrated into various platforms to enhance productivity and communication. Whether on the web or within productivity suites, Copilots provide users with intelligent assistance to streamline their tasks and improve efficiency.

Daily Tasks

Business Processes

Infra, Security, and Software Development



AI in Microsoft Azure

© Copyright KodeKloud

AI in Cloud Platforms
Microsoft Azure:
Use Copilot in the Azure portal to design, operate, optimize, and troubleshoot cloud infrastructure.

Daily Tasks

Business Processes

Infra, Security, and Software Development

The image shows two side-by-side screenshots. On the left is the Microsoft Defender Advanced hunting interface, which includes a sidebar with navigation links like Home, Incidents & alerts, Hunting (Advanced hunting selected), Custom detection rules, Actions & submissions, Threat intelligence, Learning hub, Trials, Partner catalog, Exposure management, Overview, Attack surface, Exposure insights, Secure score, Data connectors, Assets, Devices, and Metrics. The main pane displays 'Advanced hunting' with sections for Schemas, Functions, Queries, and a search bar. Below these are categories for Alerts & behaviors, Apps & identities, and Email & collaboration. On the right is the Microsoft Security Copilot interface, showing a query editor with a search bar, a results table with 3222 items, and a code editor with a PowerShell script for finding logins from domain accounts targeted by credential theft. The code is as follows:

```
Get-LogonAttempts -  
    DeviceLogonEvents  
    | Where-Object -  
        {LogonAttempts -  
            [project] Timestamp, DeviceId,  
            AccountName}
```

AI in Microsoft Copilot For Security

© Copyright KodeKloud

AI in Security Solutions

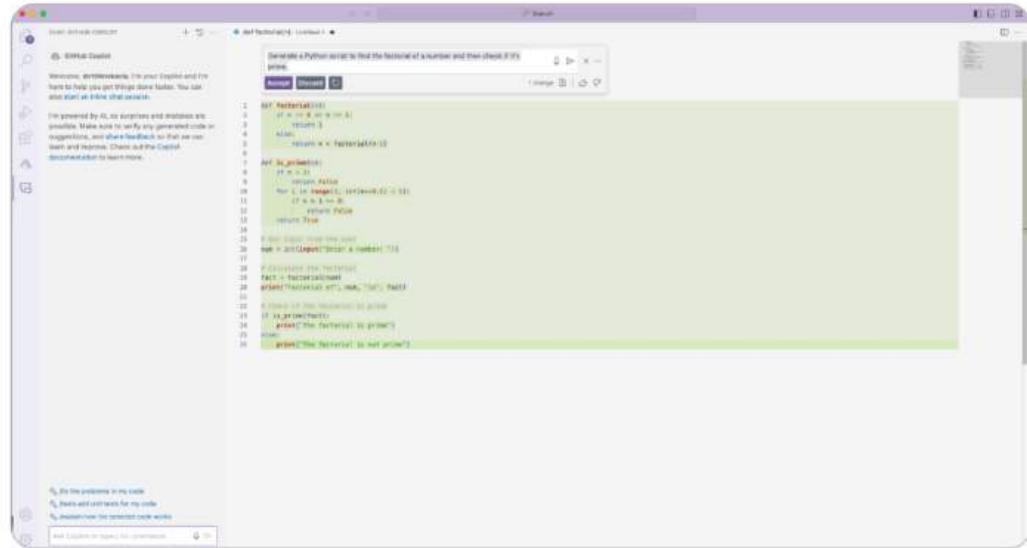
Security Copilot:

Utilize Copilot as a standalone security portal or embedded within security tools to assess and respond to security threats.

Daily Tasks

Business Processes

Infra, Security, and Software Development



AI in GitHub Copilot

© Copyright KodeKloud

AI in Software Development

GitHub Copilot:

Enhance developer productivity with Copilot by analyzing and explaining code, adding documentation, and generating new code based on natural language prompts.

Explanation:

Copilot examples showcase how generative AI can enhance IT infrastructure, security, and software development. In

Microsoft Azure, Copilot aids in managing cloud infrastructure efficiently. Security Copilot provides tools to handle security threats effectively. GitHub Copilot maximizes developer productivity by offering intelligent code assistance and documentation.

Copilot Prompts – Considerations





Draft an email to welcome new team members to the project team. Include a brief introduction of the project, highlight key team members, and provide next steps. Keep the tone friendly and professional.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

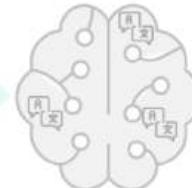
Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model



Draft an email to welcome new team members to the project team. Include a brief introduction of the project, highlight key team members, and provide next steps. Keep the tone friendly and professional.

01 Clearly state what you want the copilot to achieve.

02 Give relevant information to frame the response accurately.

03 Specify what you want to be included in the response.

04 Include additional context to ensure the response is appropriate.

05 Adjust the prompt based on the copilot's previous responses for better accuracy.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model



Draft an email to welcome new team members **to the project team**. Include a brief **introduction of the project**, highlight **key team members**, and provide next steps. Keep the tone friendly and professional.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

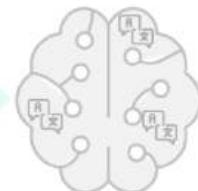
Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model

01 Clearly state what you want the copilot to achieve.

02 Give relevant information to frame the response accurately.

03 Specify what you want to be included in the response.

04 Include additional context to ensure the response is appropriate.

05 Adjust the prompt based on the copilot's previous responses for better accuracy.



Draft an email to welcome new team members to the project team. Include a brief introduction of the project, highlight key team members, **and provide next steps**. Keep the tone friendly and professional.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model

01 Clearly state what you want the copilot to achieve.

02 Give relevant information to frame the response accurately.

03 Specify what you want to be included in the response.

04 Include additional context to ensure the response is appropriate.

05 Adjust the prompt based on the copilot's previous responses for better accuracy.



Draft an email to welcome new team members to the project team. Include a brief introduction of the project, highlight key team members, and provide next steps. **Keep the tone friendly and professional.**

01 Clearly state what you want the copilot to achieve.

02 Give relevant information to frame the response accurately.

03 Specify what you want to be included in the response.

04 Include additional context to ensure the response is appropriate.

05 Adjust the prompt based on the copilot's previous responses for better accuracy.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

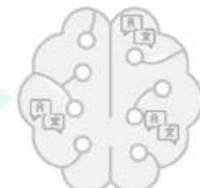
Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model



Draft an email to welcome new team members to the project team. Include a brief introduction of the project, highlight key team members, and provide next steps. Keep the tone friendly and professional.



System Message:

You are a helpful assistant...

+

Conversation History:

User: Hello

Assistant: Hello. How can I help?...

+

Current Prompt:

User: Draft an email to ...

(May be further optimized by Copilot)



Language Model

01 Clearly state what you want the copilot to achieve.

02 Give relevant information to frame the response accurately.

03 Specify what you want to be included in the response.

04 Include additional context to ensure the response is appropriate.

05 Adjust the prompt based on the copilot's previous responses for better accuracy.

Developing and Extending Copilot



Copilot Studio

Low-Code Development

Managed Service

Conversational Design

Analytics and Governance

Deployment Options

© Copyright KodeKloud

https://copilotstudio.microsoft.com/tryit

Microsoft | Microsoft Copilot Studio

Welcome

With Microsoft Copilot Studio, you can simply point your copilot to your website, and it will use generative AI to have rich, multi-turn conversations with your customers using information from your own site.

Try it now by entering your website and start talking with your very own custom copilot.

Sign up for a trial

To access the full range of Microsoft Copilot Studio capabilities — add more knowledge sources to your copilot, enjoy rich and powerful authoring capabilities using generative AI, and much more.

Chat

Hello, I'm the virtual assistant for [kodekloud.com](#). How can I help?

4 minutes ago

Which instructor created the AZ-900 course in the website?

3 minutes ago

Rithin Skaria is the instructor who created the AZ-900: Microsoft Azure Fundamentals course on the website [\[1\]](#).

1 reference

1 AZ900: Microsoft Azure Fundamentals Course | KodeKloud

3 minutes ago

Any other courses he created?

Azure AI Studio

Pro-Code Development

Infrastructure Control

Orchestration

Evaluation Tools

Flexible Deployment

Current project: lasertag-ai

Project overview

Settings

Get started

Model catalog

Model benchmarks

Prompt catalog

Project playground

Chat

Assistants PREVIEW

Images

Completions

Tools

Code PREVIEW

Prompt flow

Tracing PREVIEW

Evaluation PREVIEW

Fine-tuning PREVIEW

Components

Data

Indexes

Deployments

Content filters PREVIEW

Find the right model to build your custom AI solution

Announcements

AI21 Jamba-Instruct is here! AI21's cost-effective model optimized for long context, now available on Azure. View models Read blog

Phi-3 Serverless APIs are now available! Microsoft's latest Phi-3 SLMs offer groundbreaking performance at a small size View models Read blog

GPT-4o is now available! GPT-4o, the most advanced multimodal model from OpenAI, now available on Azure View models Read blog

All filters Collections Deployment options Inference tasks Fine-tuning tasks

Licenses

Search Models 1697

gpt-4o Chat completion

gpt-4 Chat completion

dall-e-3 Text to image

gpt-35-turbo-instruct Chat completion

davinci-002 Completions

text-embedding-ada-002 Embeddings

The screenshot shows the Azure AI Studio interface. On the left, there's a sidebar with sections for Pro-Code Development, Infrastructure Control, Orchestration, Evaluation Tools, and Flexible Deployment. The main area is titled 'Find the right model to build your custom AI solution'. It features three announcements: 'AI21 Jamba-Instruct is here!', 'Phi-3 Serverless APIs are now available!', and 'GPT-4o is now available!'. Below the announcements is a search bar and a 'Models 1697' button. The main content area displays a grid of AI models, each with a preview icon, name, and description. The models shown are gpt-4o (Chat completion), gpt-4 (Chat completion), dall-e-3 (Text to image), gpt-35-turbo-instruct (Chat completion), davinci-002 (Completions), and text-embedding-ada-002 (Embeddings).

Generative AI

Fundamentals of Azure OpenAI

Module Introduction

01 Exploring Azure OpenAI

02 Models Supported by Azure OpenAI

03 Azure OpenAI Capabilities

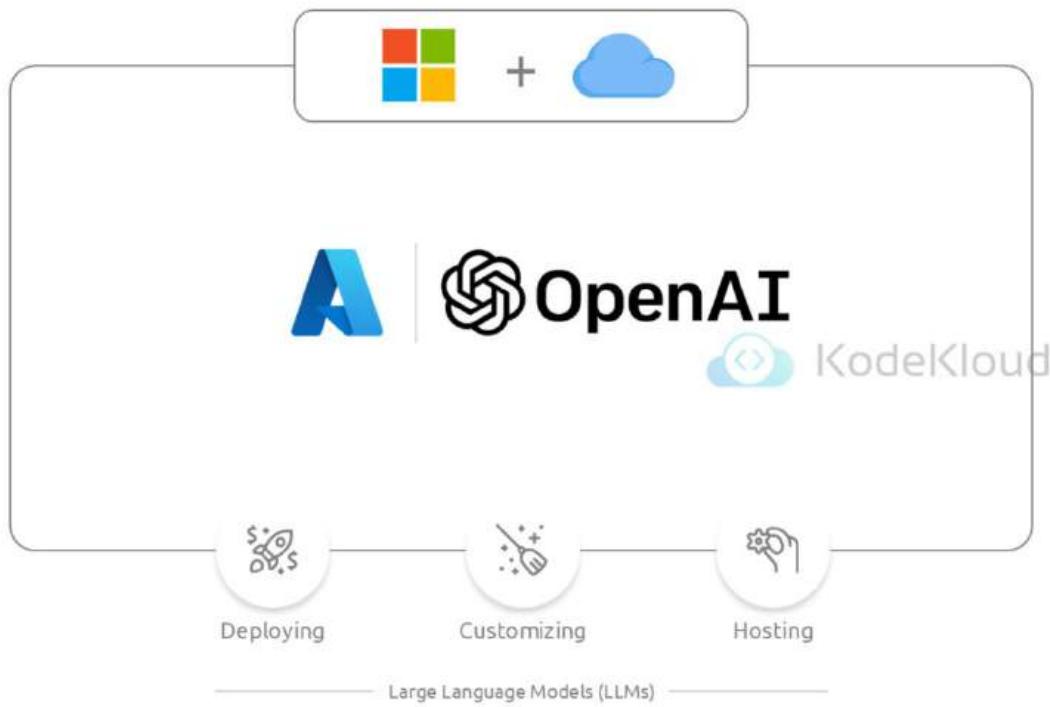
04 Azure OpenAI Services



KodeKloud

Exploring Azure OpenAI







01

Pre-trained AI
models

02

Model
customization

03

Responsible AI
tools

04

KodeKloud



Enterprise
security features



OpenAI

Development Methods

Azure AI Studio

REST API

Supported SDKs

Azure CLI

Models Supported by Azure OpenAI





Supports LLMs

GPT-4 and GPT-4 Turbo

GPT-4

GPT-3.5

Embeddings

DALL-E (preview)

Azure OpenAI Capabilities



How to get started with



Azure AI | Azure OpenAI Studio

There's more to discover in the updated Azure AI Studio

Try more models, capabilities, and tools for building your AI solution, or bring your existing deployments without any extra steps.

Explore Azure AI Studio

Deploy GPT-4o: GPT-4o is now available in many regions. Deploy and try it in the chat playground. [Learn more](#) [Go to deployments](#)

Azure OpenAI Studio [Privacy & cookies](#)

Welcome to Azure OpenAI service

Explore the generative AI models, craft unique prompts for your use cases, and fine-tune select models.

Get started

Chat playground

Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.

Try it now

Assistants playground

PREVIEW

Speed up development of GPT-powered AI Assistants with prebuilt conversation state management and customization tools.

Try it now

Text generation



Text generation



© Copyright KodeKloud



There's more to discover in the updated Azure AI Studio

Try more models, capabilities, and tools for building your AI solution, or bring your existing deployments without any extra steps.

[Explore Azure AI Studio](#)

Deploy GPT-4o: GPT-4o is now available in many regions. Deploy and try it in the chat playground. [Learn more](#) [Go to deployments](#)

Azure OpenAI Studio

[Privacy & cookies](#)

Welcome to Azure OpenAI service

Explore the generative AI models, craft unique prompts for your use cases, and fine-tune select models.

Get started



Chat playground

Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.

[Try it now](#)



Assistants playground

Speed up development of GPT-powered AI Assistants with prebuilt conversation state management and customization tools.

[Try it now](#)



Playgrounds

Experimentation

Configuration

Natural Language Capabilities

Generative pre-trained transformer (GPT) models excel in both understanding and creating natural language.

Prompt: Create a 3-day travel itinerary for visiting Paris, including major attractions and dining options.

Response:

Itinerary:

Day 1:

1. Morning: Visit the Eiffel Tower and take an elevator to the top.
2. Lunch: Enjoy a meal at Le Jules Verne located within the Eiffel Tower.
3. Afternoon: Explore the Louvre Museum and see the Mona Lisa.
4. Dinner: Dine at Le Comptoir du Relais.

Day 2:

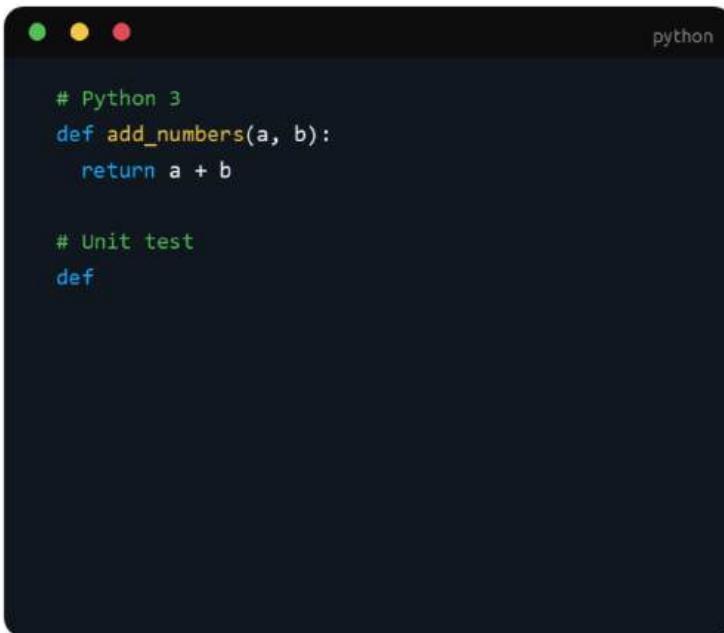
1. Morning: Walk through the beautiful gardens of Luxembourg.
2. Lunch: Try traditional French cuisine at Café de Flore.
3. Afternoon: Visit the Notre-Dame Cathedral and climb to the bell towers.
4. Dinner: Enjoy a fine dining experience at Le Meurice.

Day 3:

1. Morning: Stroll along the Champs-Élysées and visit Arc de Triomphe.
2. Lunch: Have lunch at Ladurée for some famous macarons.
3. Afternoon: Take a boat cruise along the Seine River.
4. Dinner: End your trip with a dinner at L'Astrance.

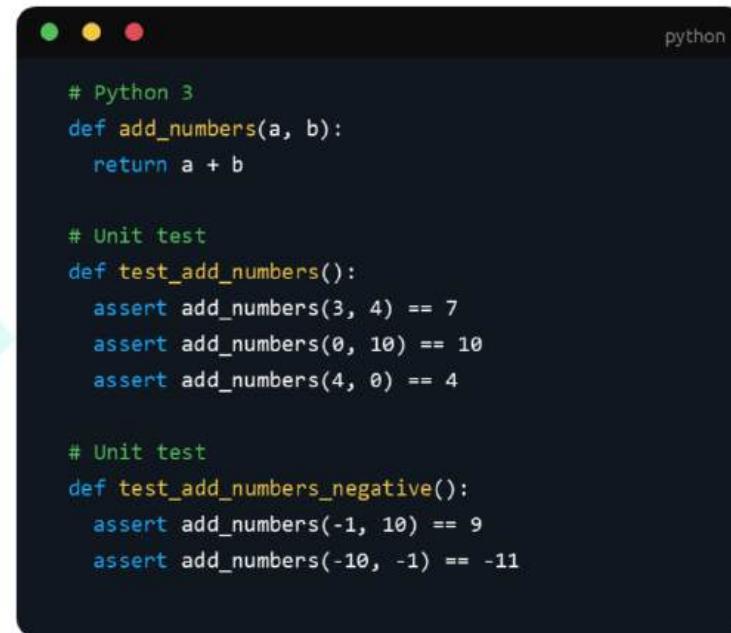
Code Generation Capabilities

PT models transform natural language or code snippets into functional code.



```
# Python 3
def add_numbers(a, b):
    return a + b

# Unit test
def
```



```
# Python 3
def add_numbers(a, b):
    return a + b

# Unit test
def test_add_numbers():
    assert add_numbers(3, 4) == 7
    assert add_numbers(0, 10) == 10
    assert add_numbers(4, 0) == 4

# Unit test
def test_add_numbers_negative():
    assert add_numbers(-1, 10) == 9
    assert add_numbers(-10, -1) == -11
```

Image Generation Capabilities



Generative AI models can
edit and create images.

Image Generation Capabilities



Image Generation Capabilities



DALL·E

Create



KodeKloud

Image Generation Capabilities



[Edit](#)



KodeKloud

Image Generation Capabilities



Variation



KodeKloud

Prompt: Create two variations of an image of an ant singing



Responsible Generative AI

Module Introduction

01

Plan a Responsible Generative AI Solution



KodeKloud

Plan a Responsible Generative AI Solution





Scenario: Implementing an AI-powered customer service chatbot

- Biases
- Offensive content
- Misinformation

- Evaluate chatbot interactions

- Implement filters
- Guidelines

- Monitor and update chatbot

© Copyright KodeKloud

Identify

Recognize potential harms that are relevant to your planned solution.

Measure

Assess the presence of these harms in the outputs generated by your solution.

Mitigate

Address the harms at multiple layers in your solution to minimize their presence and impact.

Operate

Deploy and maintain the solution responsibly by defining and following an operational readiness plan.

Example:

Scenario: Implementing an AI-powered customer service chatbot

Identify: Potential harms include biases in responses, inappropriate or offensive content, and misinformation.

Measure: Regularly evaluate chatbot interactions to identify instances of bias, inappropriate content, or misinformation.

Mitigate: Implement filters and guidelines to ensure the chatbot provides accurate, respectful, and unbiased responses.

Operate: Continuously monitor and update the chatbot, ensuring it adheres to ethical standards and provides a reliable service



KodeKloud

© Copyright KodeKloud

Visit www.kodekloud.com to learn more.