

MOBIUS

Tomorrow, Delivered.

**The Platform Academy
AI & ML Workshop**

Fueling Growth with Training Solutions of Tomorrow.



Who am I?

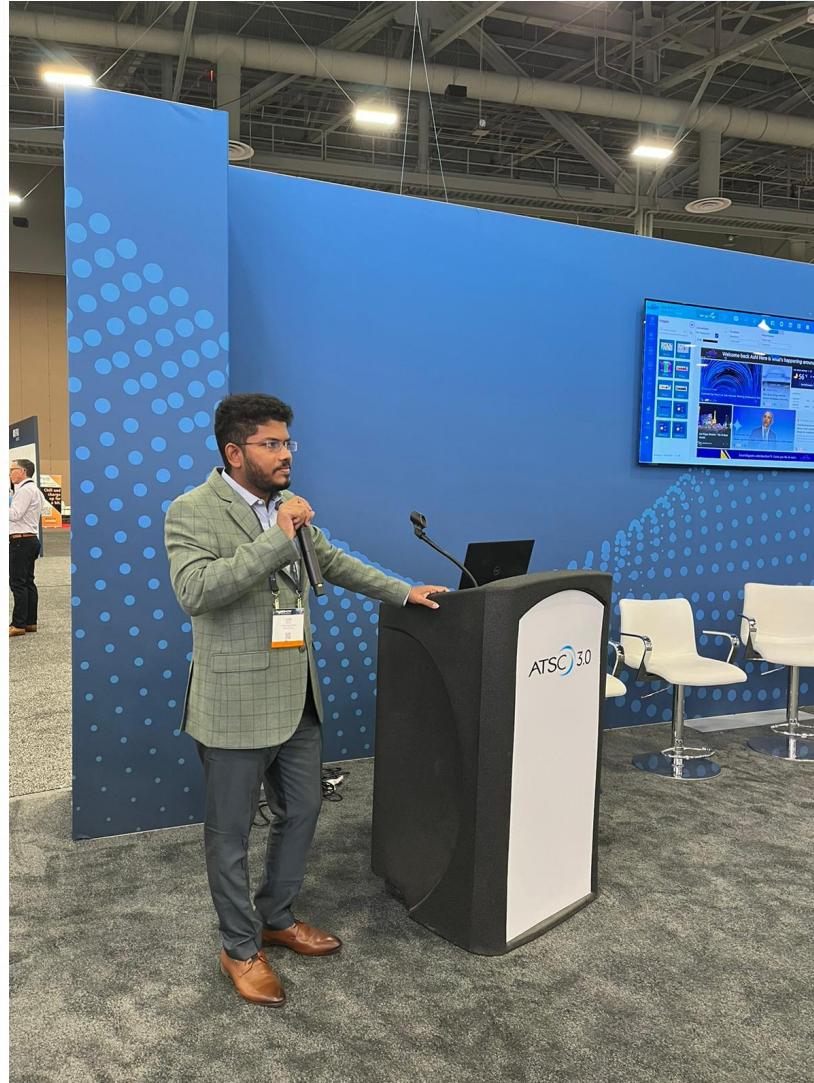
Jose Anitt

Platform Architect, Tech Enthusiast

IIT Bombay, 2014 Grad

<https://www.linkedin.com/in/gjanitt/>

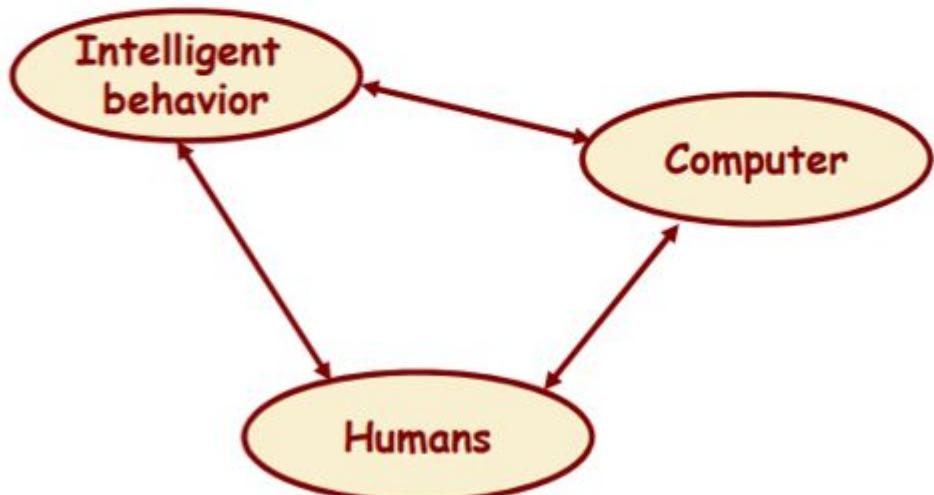
Building Mobius

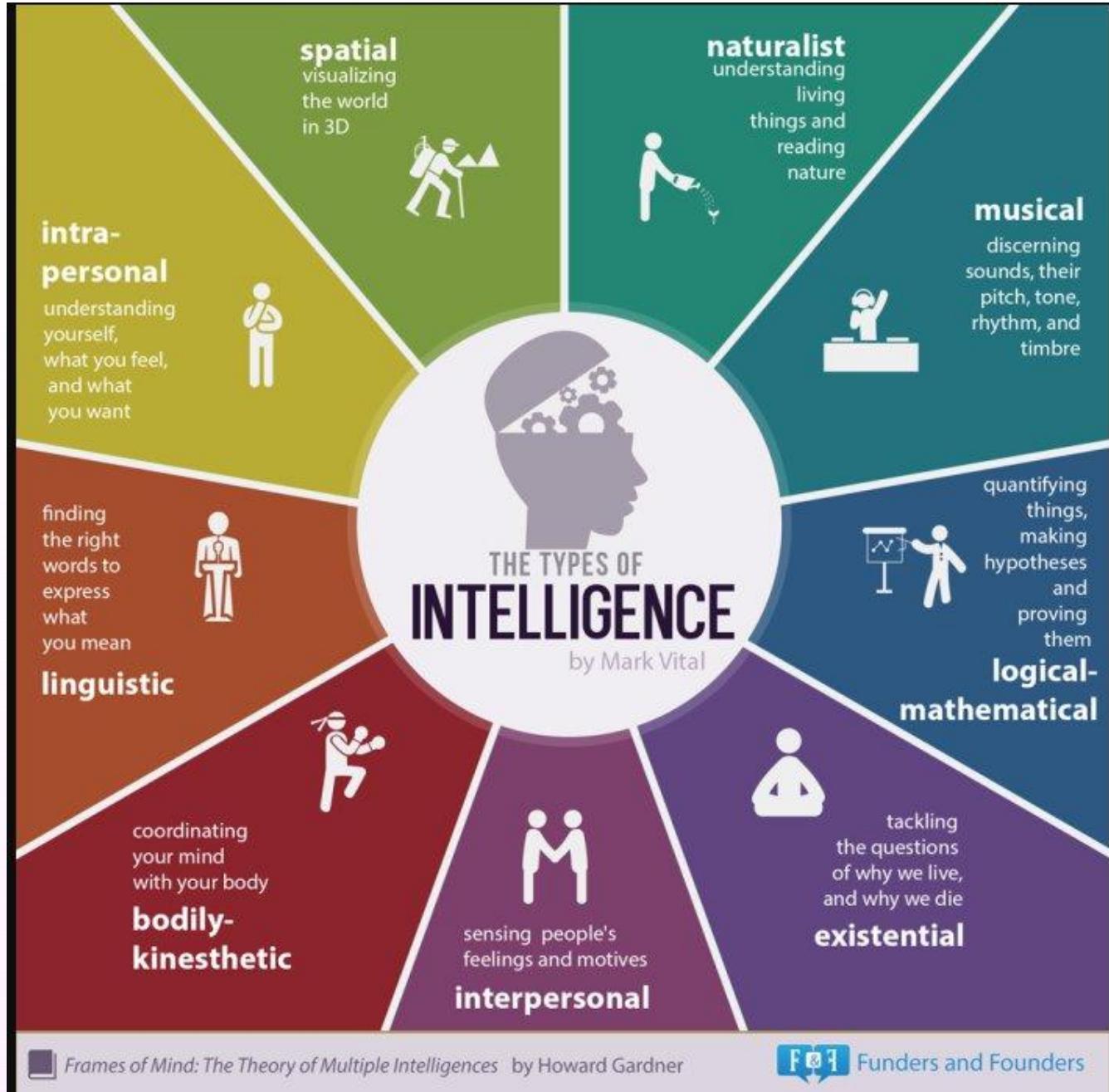


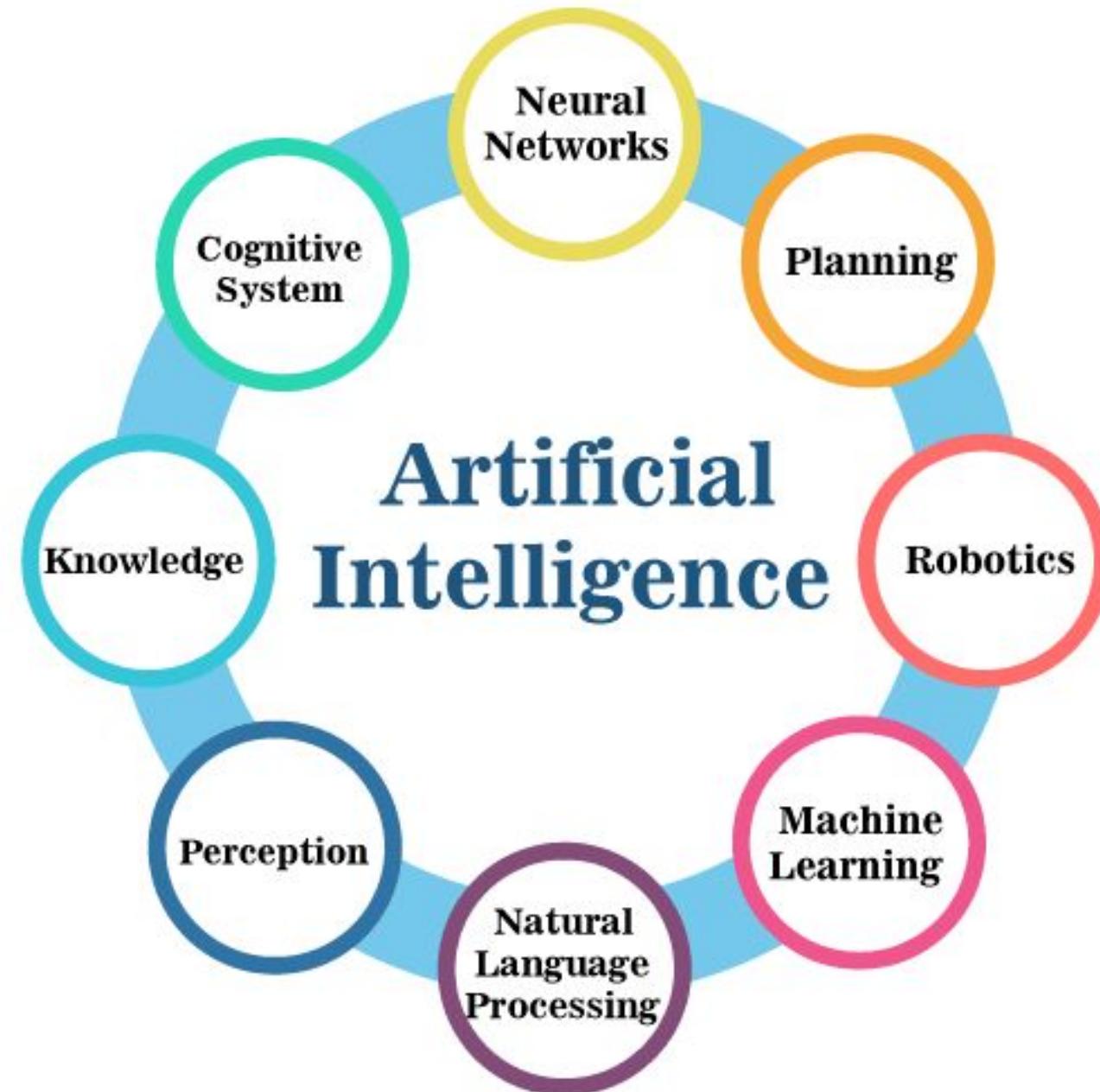
What is AI?

an attempt of

- AI is the reproduction of human reasoning and intelligent behavior by computational methods







What is ML?

- "Field of study that gives computers the ability to learn without being explicitly programmed" - Arthur Samuel (1959)
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" - Tom M. Mitchell (1998)

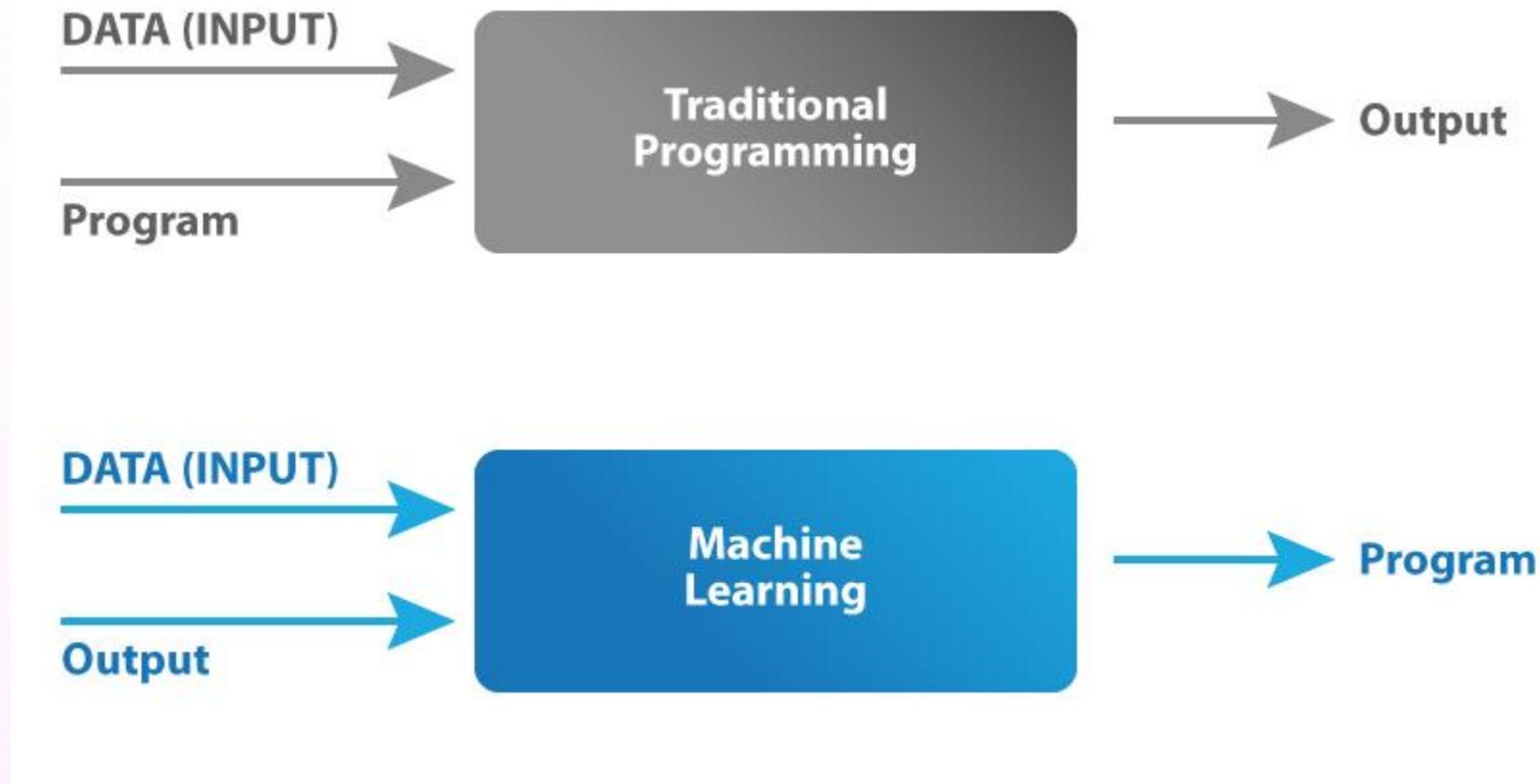
Example: Spam filtering

E: Watching a mail being labelled as spam or not

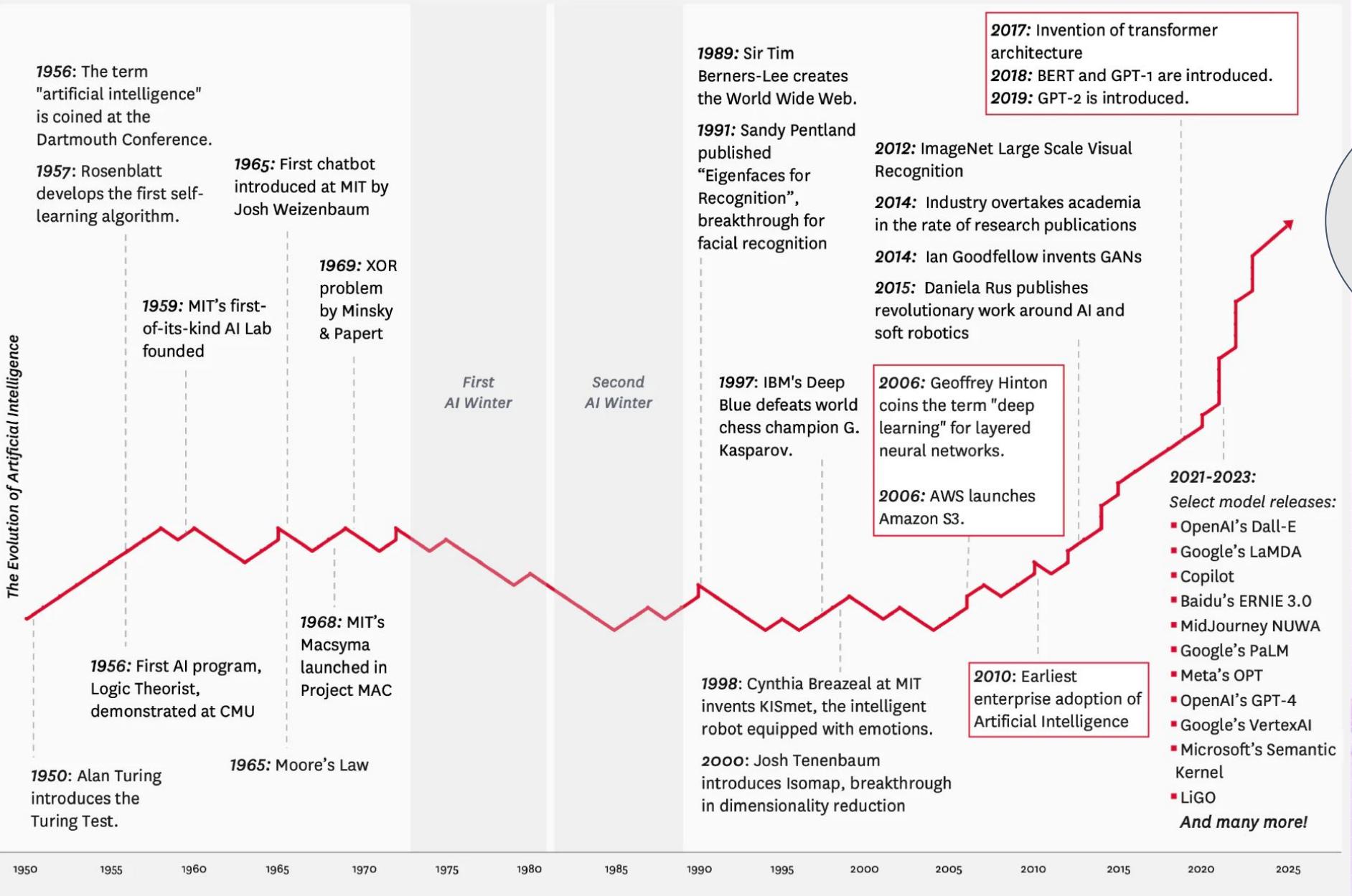
T: Classifying an email as spam or not

P: Fraction of mails correctly classified as spam

What is ML?



Glasswing Ventures' proprietary, data-driven timeline of the evolution of Artificial Intelligence



History and Evolution of AI



IBM Deep Blue

Computer that defeated world chess champion Garry Kasparov in 1997



Types of ML Algorithms

Supervised Learning: Learning from labeled data to predict outcomes.

Unsupervised Learning: Finding hidden patterns in unlabeled data.

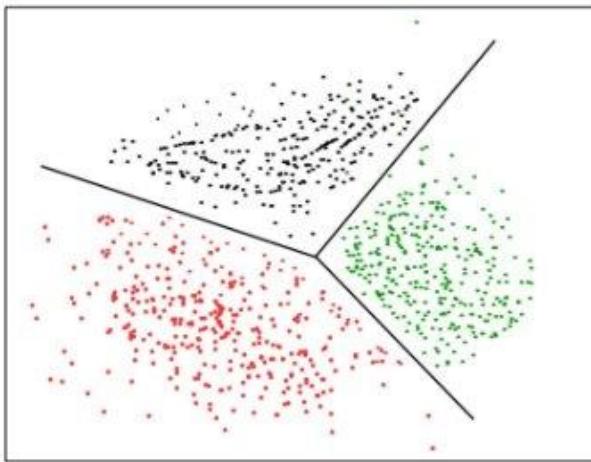
Reinforcement learning: Learning optimal actions through trial and error.



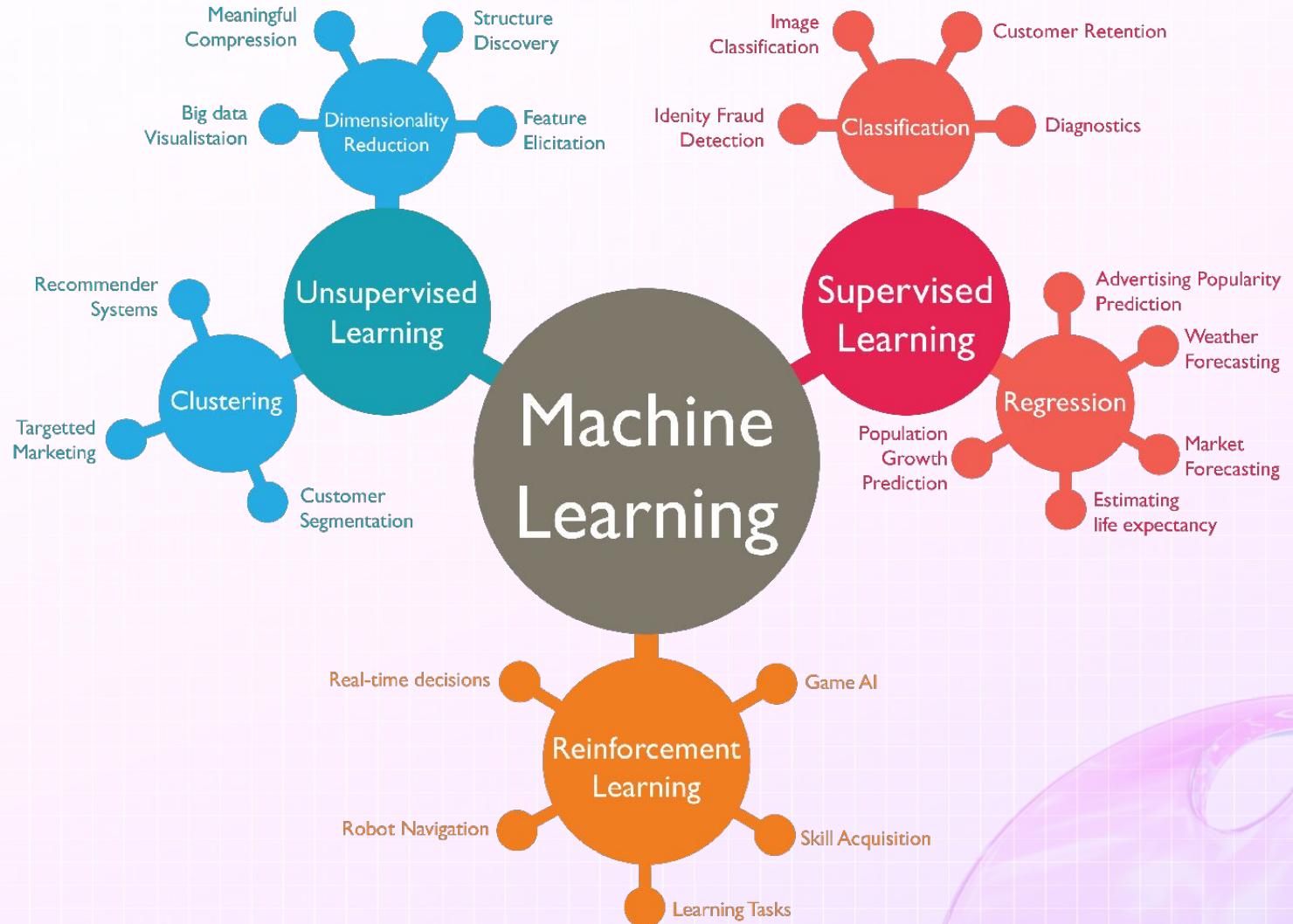
Types of ML Algorithms



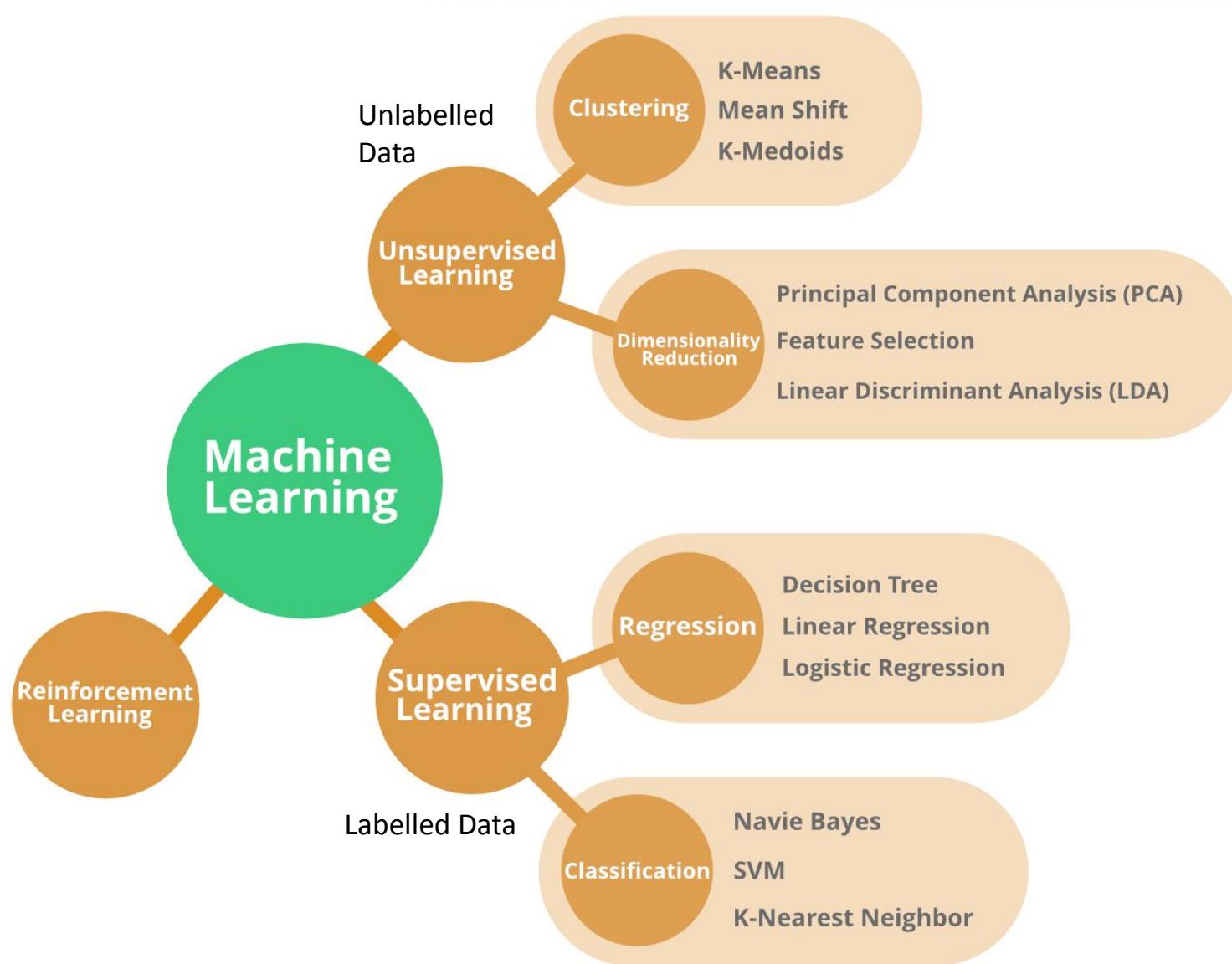
CAT



Types of ML Algorithms



Popular ML Algorithms



Discrete
Input Data -> to find **Input Regularities**

Continuous
Input Data -> to find the best **lower dimensional representation**

Continuous and Quantitative
Input Data -> Output is a **Number**

Discrete and Qualitative
Input Data -> Output is a **Class**

Use Cases of AI & ML

It's already all around you

Search engine results

Email spam filters

Amazon & Netflix recommendations

Twitter & Instagram - personalized feeds

Personalized music - spotify

IoT - Home Automation

Tesla cars - autonomous drives

Surveillance cameras - safety

Supermarket stores - stocking orders

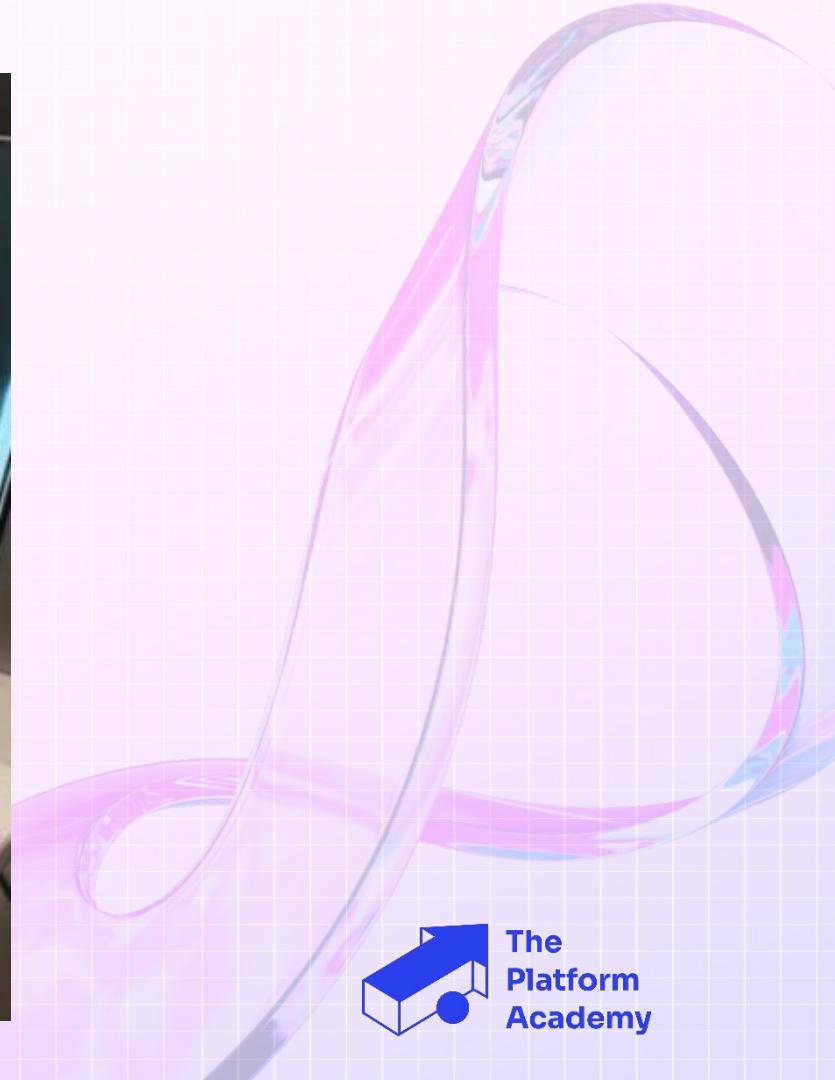
Credit card approvals & offers

Future

Addressing bias, privacy, and ethical considerations

More Use Cases

Wearable Device Allows Humans To Control Robots with Brain Waves



More Use Cases

SORA (Open-AI)

Sora is an advanced text-to-video AI model developed by OpenAI.

This model allows users to input textual descriptions, which Sora then transforms into short, high-quality video clips up to a minute long.

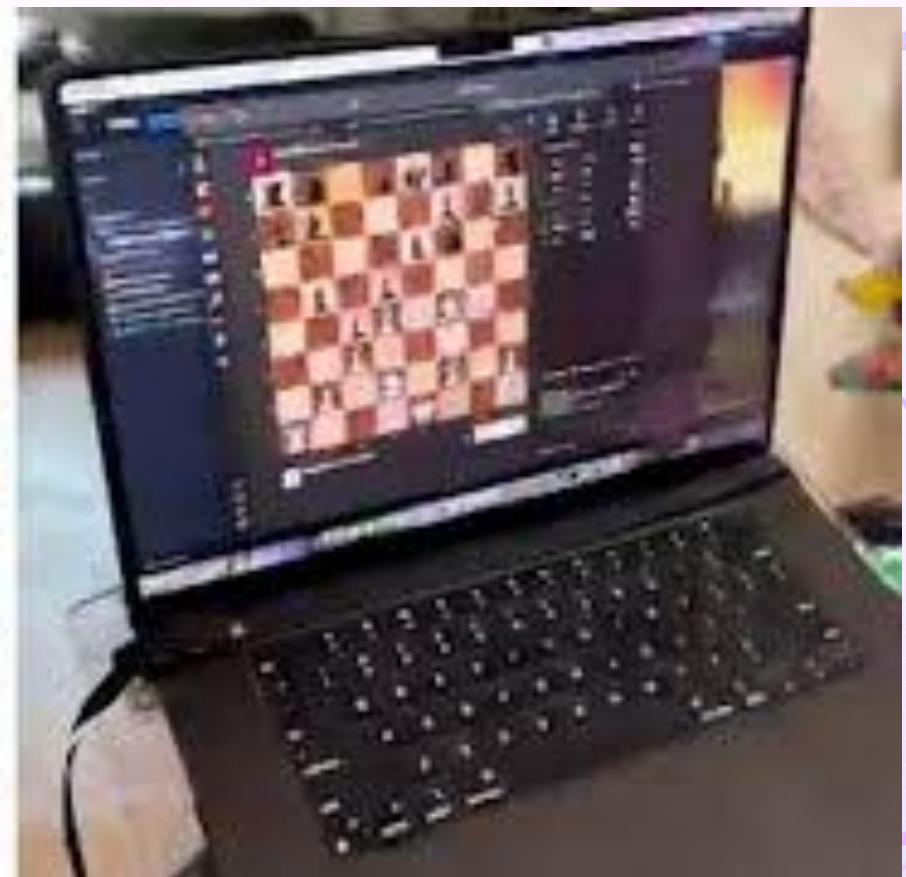


More Use Cases

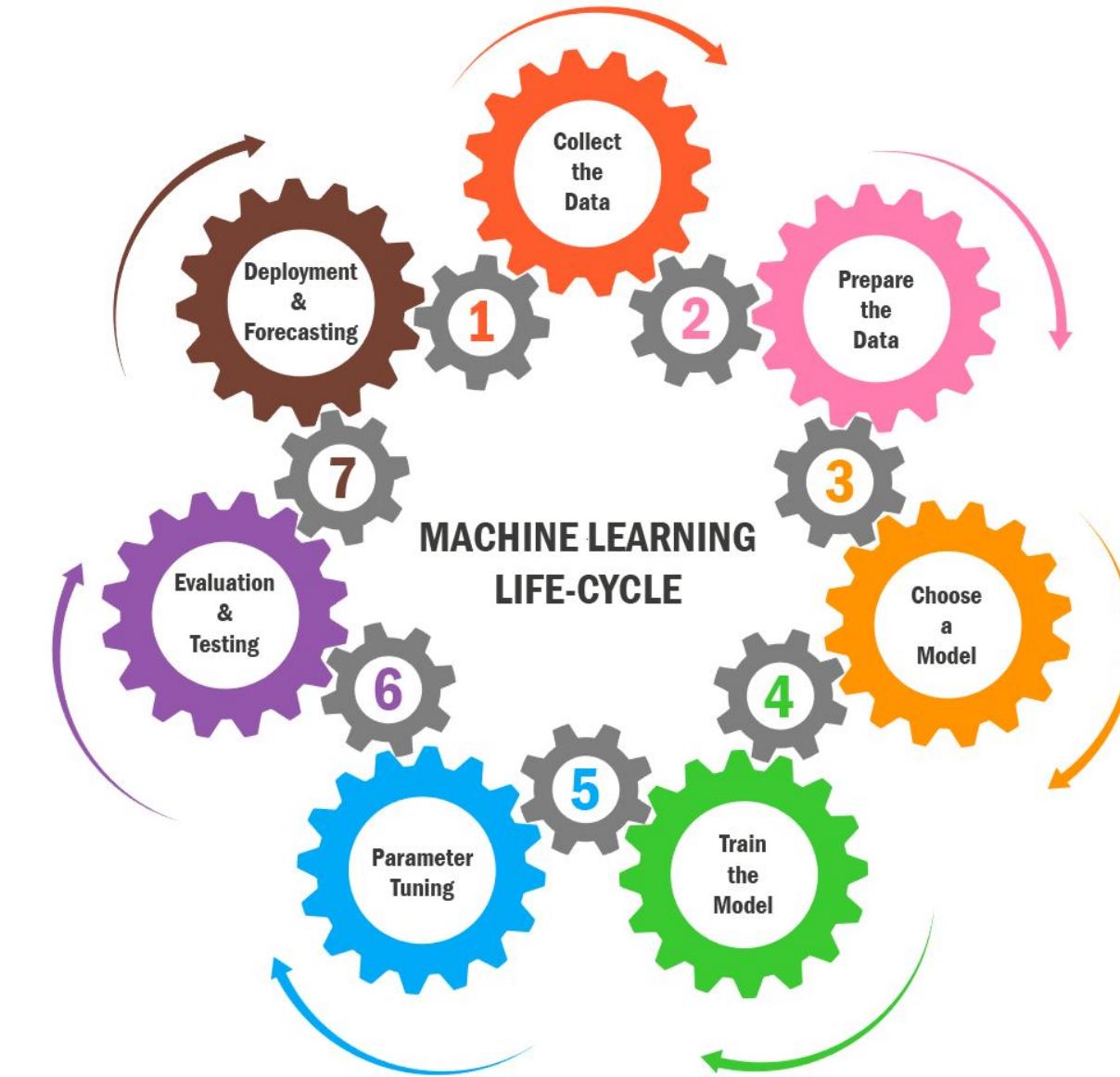
NEURALINK

Neuralink is a technology company building a device “designed to connect human brains directly to computers”

The implant itself is called “the Link.” This coin-sized brain chip is surgically embedded under the skull, where it receives information from neural threads that fan out into different sections of a subject’s brain in control of motor skills.



ML Model Life Cycle



Understanding ML Workflow

Data Collection

The first step in the ML workflow involves gathering relevant data from various sources. The quality and quantity of data significantly impact the model's performance.

Sources - databases, web scraping, public datasets

Data Preprocessing

Preparing the collected data for analysis by cleaning, normalizing, and transforming it. This step ensures that the data is in a suitable format for training the model.

Cleaning - handling missing values and outliers

Feature Scaling - Normalization and Standardization

Feature Engineering - handling categorical data

Understanding ML Workflow

Model Training

Feeding the preprocessed data into a machine learning algorithm to learn patterns and relationships. This step involves selecting an appropriate model and tuning its parameters.

- Splitting data into training and validation sets.
- Feeding training data into the model to learn patterns.
- Tuning hyperparameters to optimize performance.

Model Evaluation

Assessing the trained model's performance using a separate test set to ensure it generalizes well to new, unseen data.

- **Accuracy:** Proportion of correctly predicted instances.
- **Precision & Recall:** Measures for evaluating classification models.
- **Mean Squared Error (MSE):** Commonly used for regression models.

Understanding ML Workflow

Model Deployment

Integrating the trained model into a real-world application where it can make predictions on new data.

- API integration
- Embedding models into software applications
- Cloud deployment

Data Preprocessing - Normalization & Standardization

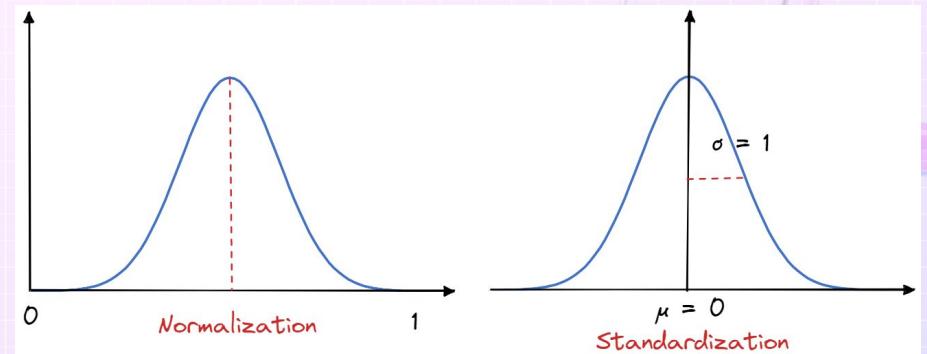
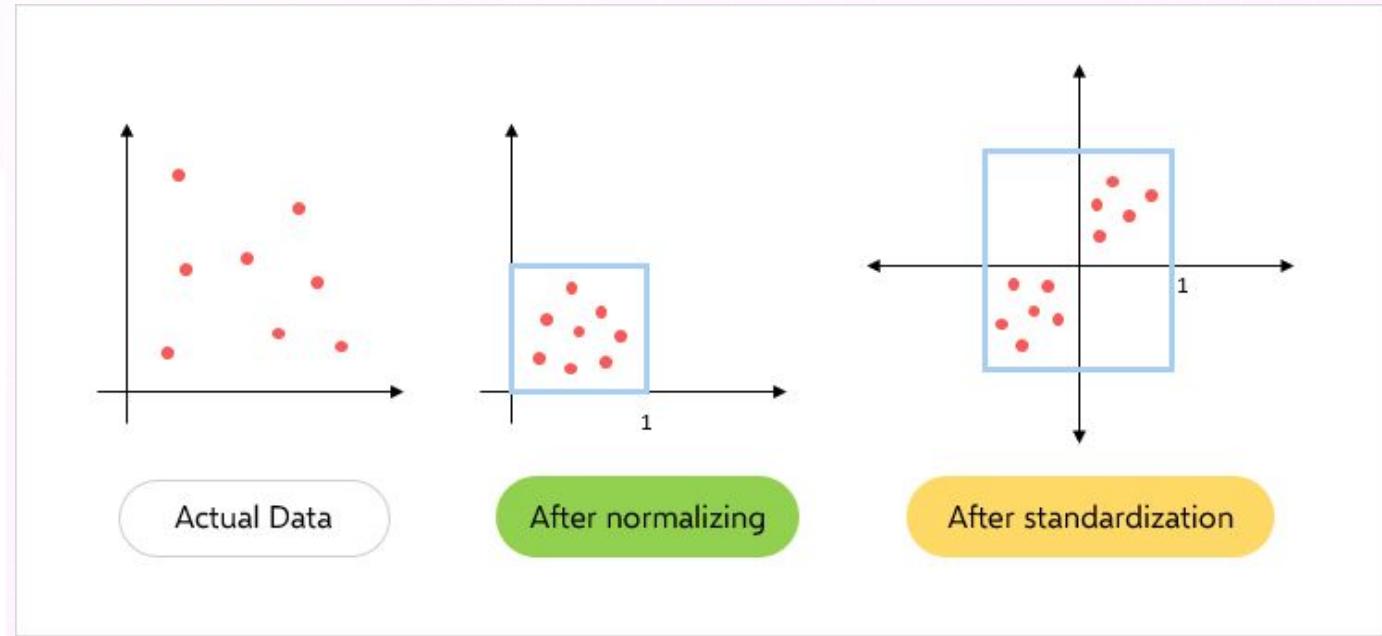
Feature scaling

Normalization

Standardization

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

$$X' = \frac{X - \text{Mean}}{\text{Standard deviation}}$$



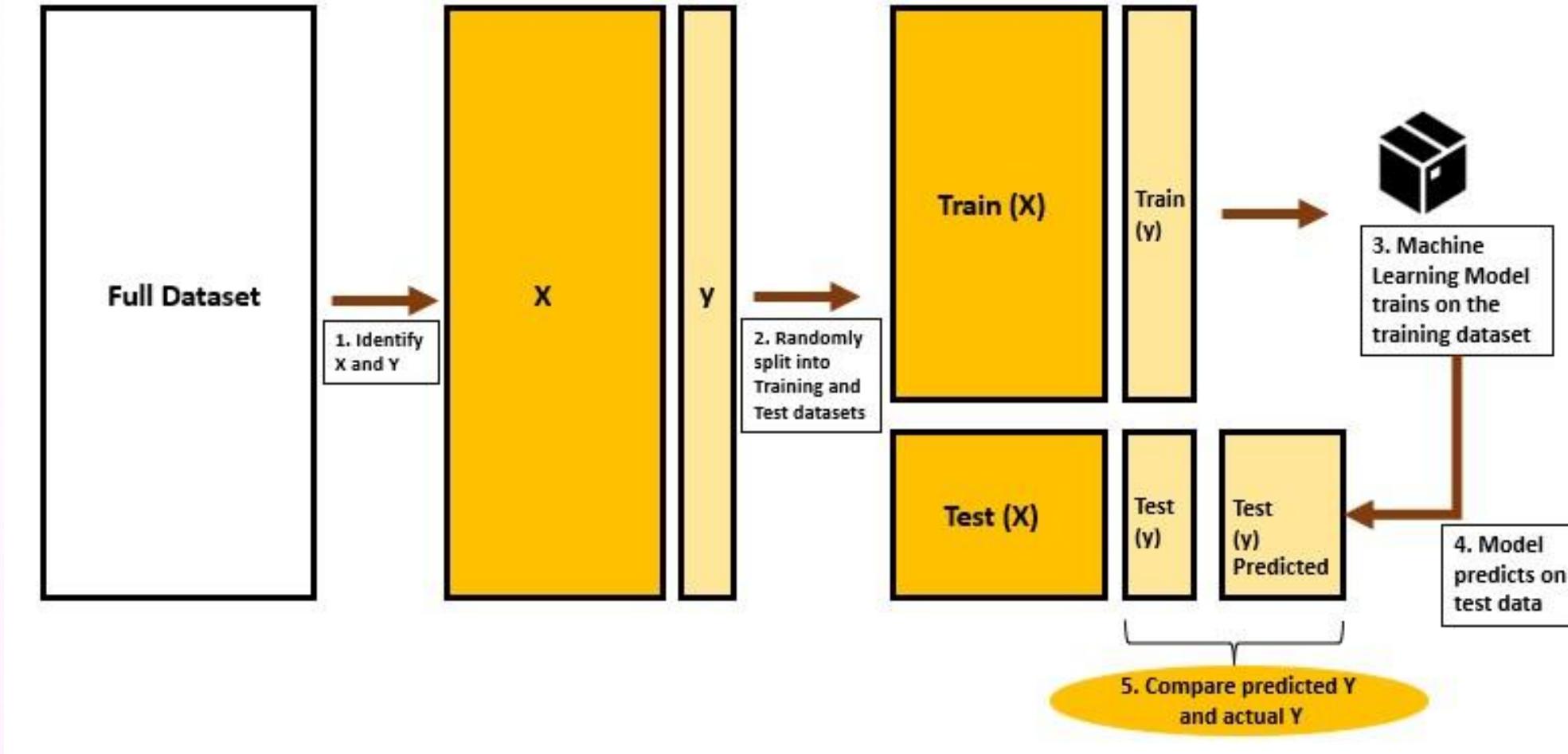
Data Preprocessing - One Hot Encoding

id	color
1	red
2	blue
3	green
4	blue

One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Model Training - Train Test Data Split



Model Evaluation - Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	True positive	False negative
	Negative	False positive	True negative

Model Evaluation - Precision and Recall

Precision - "Of all the instances the model predicted as positive, how many were actually positive?"

High precision indicates that when the model predicts a positive result, it is usually correct. However, it does not consider how many actual positive instances were missed (that's where recall comes in).

Recall - "Of all the actual positive instances, how many did the model correctly identify as positive?"

High recall means the model is good at capturing most of the actual positive cases, but it doesn't take into account how many of the predicted positives were actually false.

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Model Evaluation - Precision vs Recall Trade-off

High Precision, Low Recall

This indicates the model is very selective and only classifies instances as positive when it's very confident. However, it may miss many actual positives (false negatives).

Needed when the cost of a false positive is high. For example, a spam filter should have high precision to avoid marking important emails as spam.

High Recall, Low Precision

This suggests the model is less strict, trying to capture all possible positive cases, but this may include many false positives.

Needed when the cost of missing a positive instance is high. For example, in disease screening tests, high recall is crucial to identify all possible cases of the disease.

Balance = F1 Score = harmonic mean of precision and recall

Model Evaluation - Confusion Matrix - More

	Predicted Positive	Predicted Negative	
Actual Positive	TP <i>True Positive</i>	FN <i>False Negative</i>	Sensitivity $\frac{TP}{(TP + FN)}$
Actual Negative	FP <i>False Positive</i>	TN <i>True Negative</i>	Specificity $\frac{TN}{(TN + FP)}$
	Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Python for ML - Libraries and Tools

Why Python?

- Simple
- Popular
- Extensive Support
- Rich ecosystem of tools and libraries
- Large Community



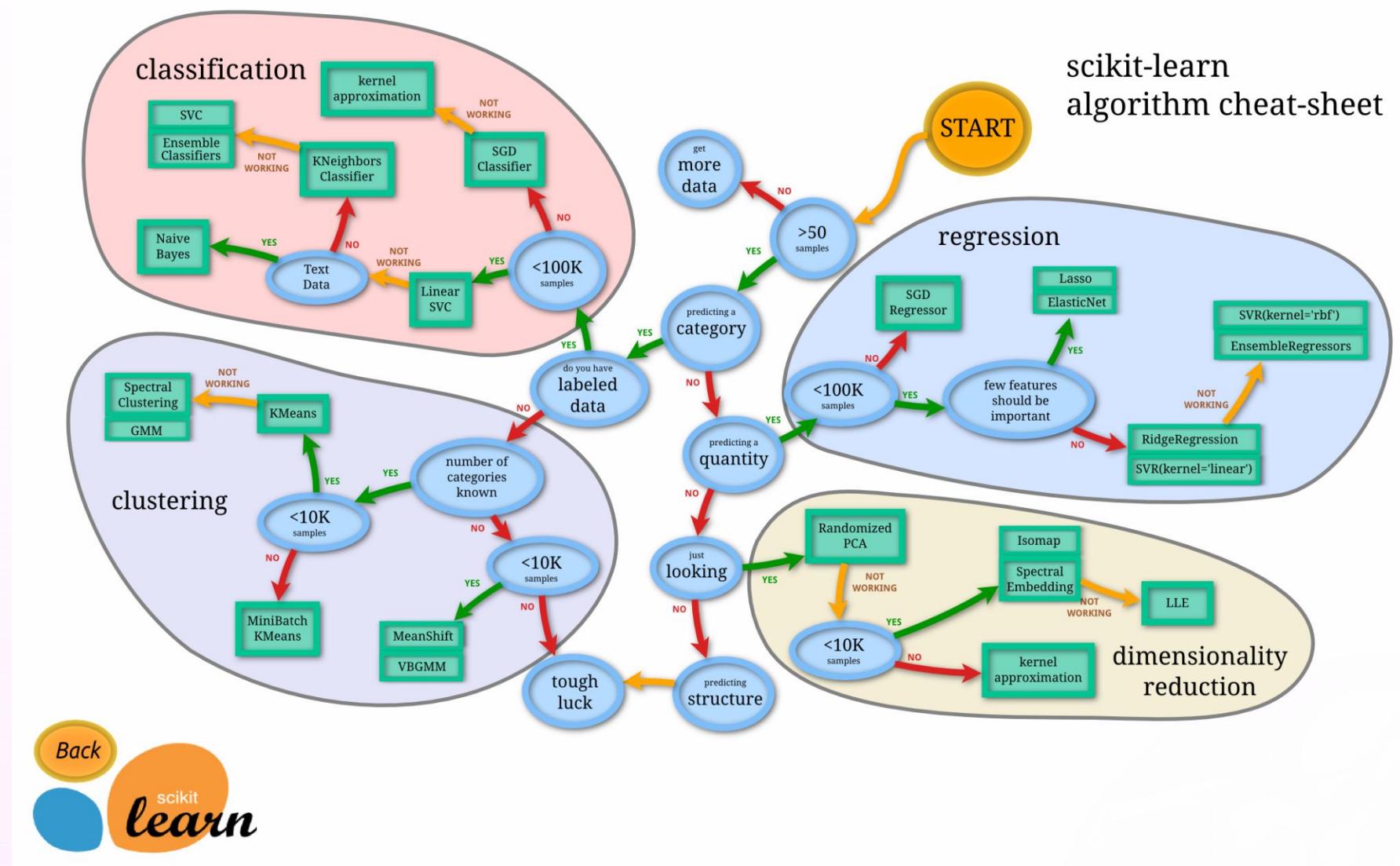
Python for ML - Scikit-learn

A widely used library for machine learning in Python, scikit-learn provides simple and efficient tools for data analysis and modeling.

Features

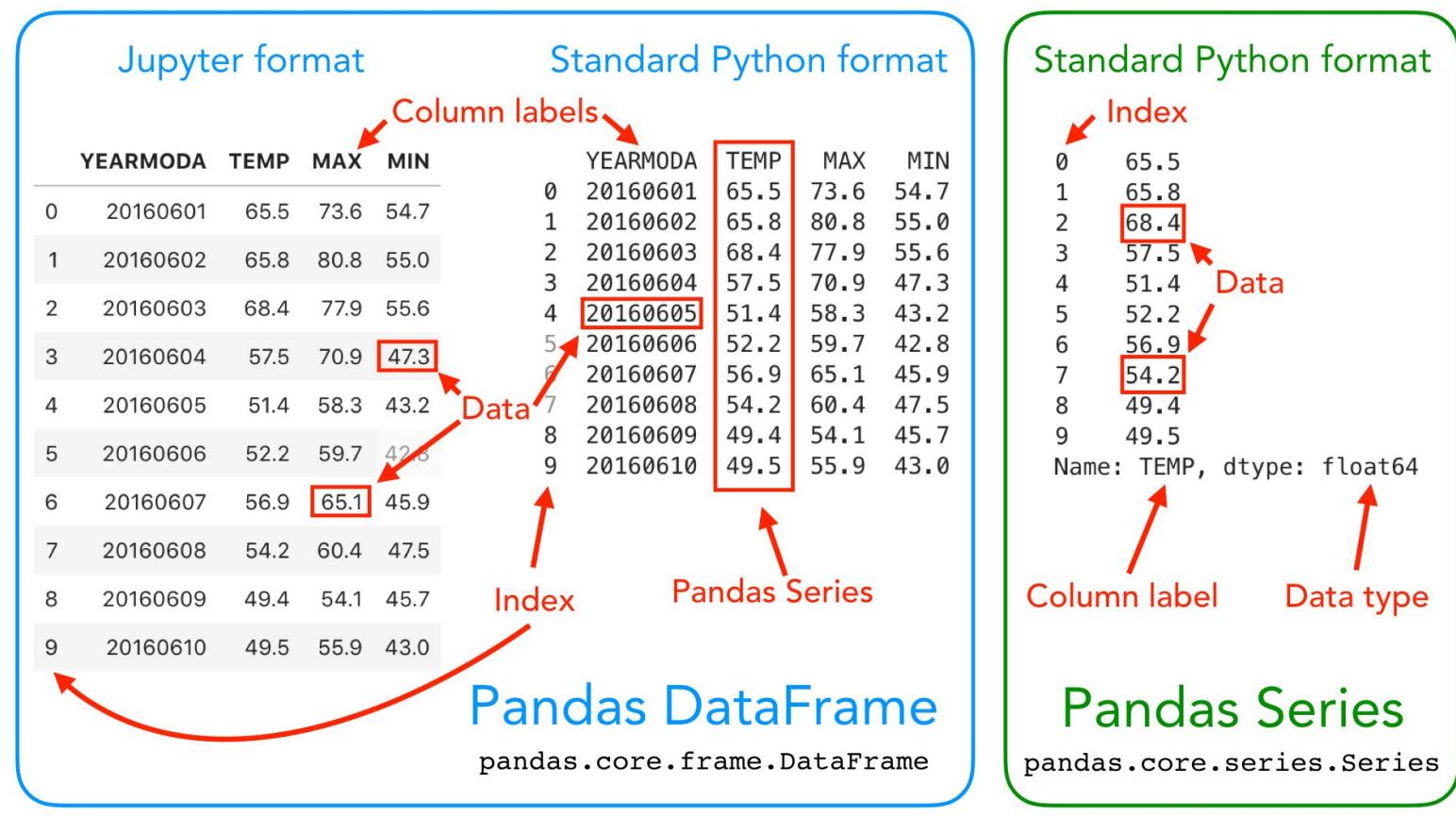
- Classification (e.g., logistic regression, support vector machines)
- Regression (e.g., linear regression)
- Clustering (e.g., k-means)
- Dimensionality Reduction (e.g., PCA)
- Model Evaluation (e.g., cross-validation)

Python for ML - Scikit-learn



Python for ML - Pandas

A powerful data manipulation and analysis library that provides data structures like DataFrames, making it easy to handle and analyze structured data.



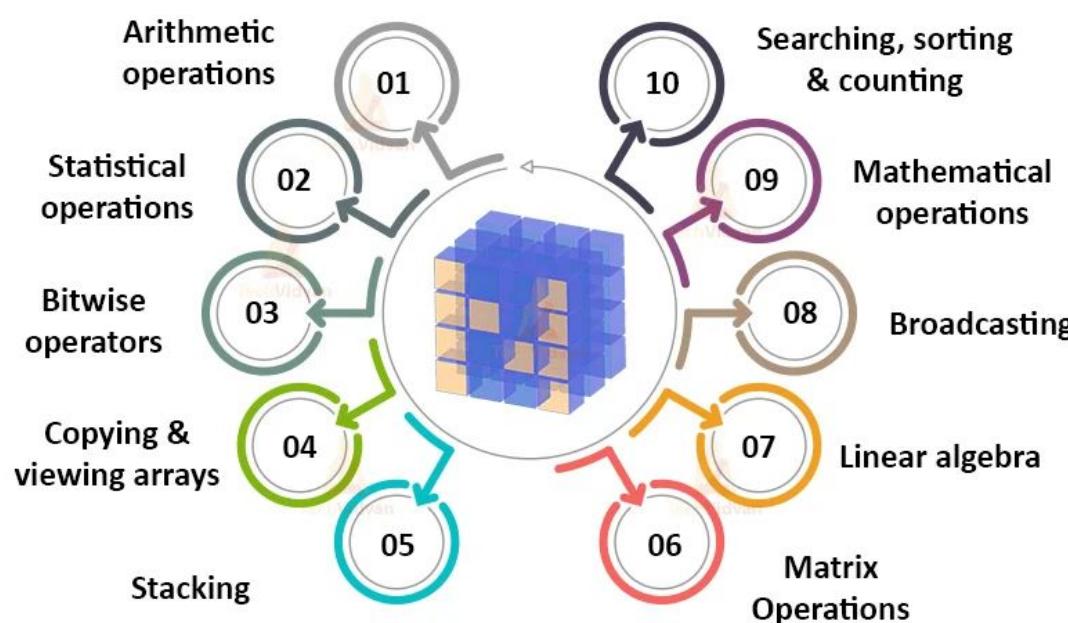
Features:

- Data cleaning and preparation
- Data manipulation and analysis
- Handling missing data

Python for ML - NumPy

NumPy is the foundational package for numerical computing in Python, offering support for arrays, matrices, and a collection of mathematical functions.

Uses of NumPy



Features:

- Fast array processing
- Mathematical operations (e.g., linear algebra)

Python for ML - Other Useful Libraries

TensorFlow

Open-source framework for deep learning and complex numerical computations

Keras

High-level neural networks API, running on top of TensorFlow, making it easier to build and train models.

Matplotlib & Seaborn

Libraries for data visualization, providing tools to create plots, graphs, and charts for exploratory data analysis

Jupyter Notebooks

An interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text.

Linear Regression Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Generate new synthetic data for house sizes (in square feet) and prices (in $1000)
np.random.seed(42) # Changed seed for different data generation
sizes = np.random.rand(100, 1) * 3000 + 1000 # Sizes between 1000 and 4000 sqft
prices = 50 + 0.2 * sizes + np.random.randn(100, 1) * 20 # Prices in $1000 with some noise

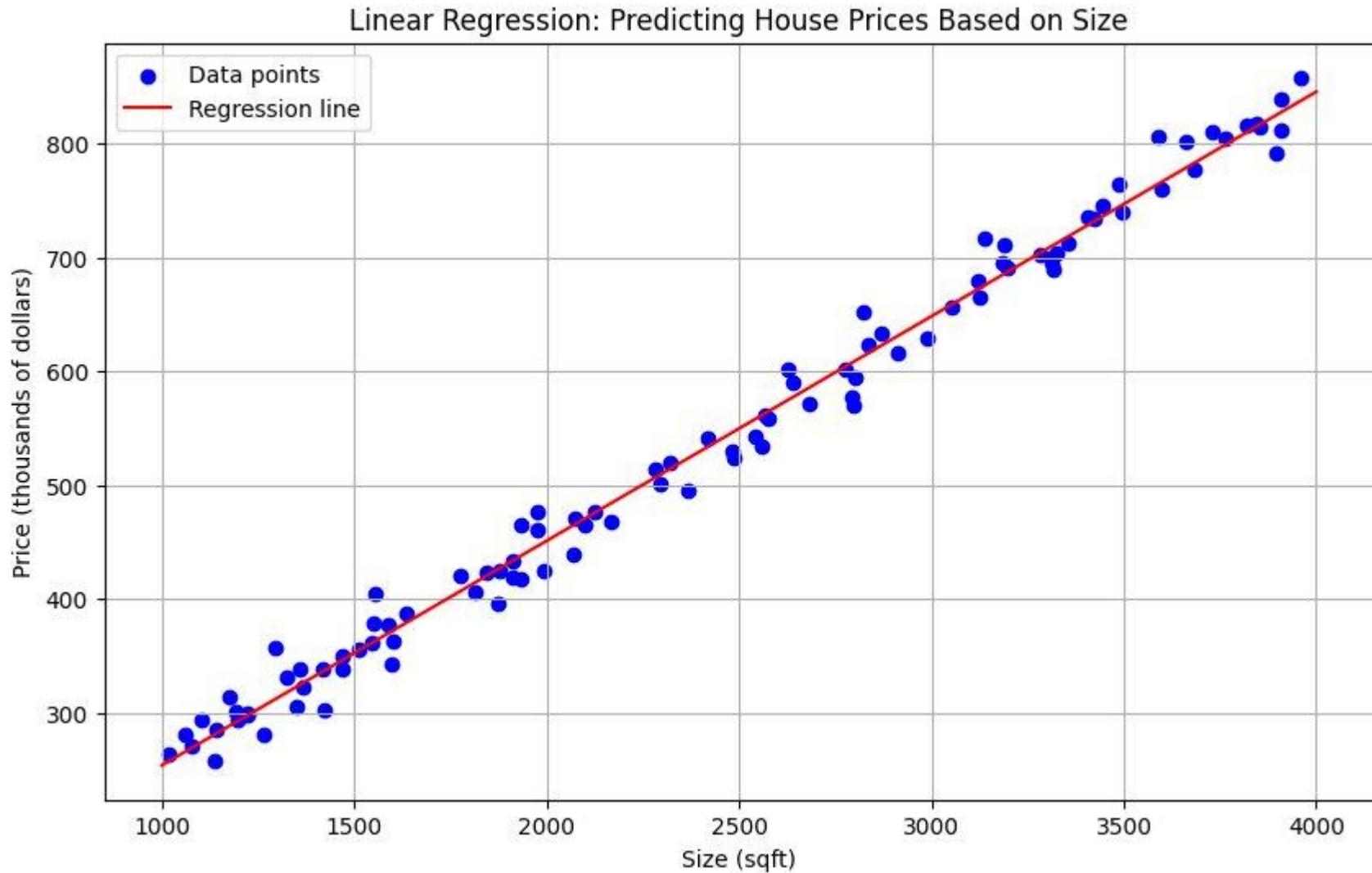
# Fit the linear regression model
model = LinearRegression()
model.fit(sizes, prices)

# Generate predictions for the range of sizes
sizes_new = np.linspace(1000, 4000, 100).reshape(-1, 1)
prices_predict = model.predict(sizes_new)

# Plot the data and the regression line
plt.figure(figsize=(10, 6))
plt.scatter(sizes, prices, color="blue", label="Data points")
plt.plot(sizes_new, prices_predict, color="red", label="Regression line")
plt.xlabel("Size (sqft)")
plt.ylabel("Price (thousands of dollars)")
plt.title("Linear Regression: Predicting House Prices Based on Size")
plt.legend()
plt.grid(True)
plt.show()
```

House Price Prediction

Linear Regression Code Output



Time for
Google Colab
NumPy, Pandas
Linear Regression



Linear Regression

It tries to find out the best linear relationship that describes the data

It assumes that there exists a linear relationship between a dependant variable (continuous value) and independent variable(s)

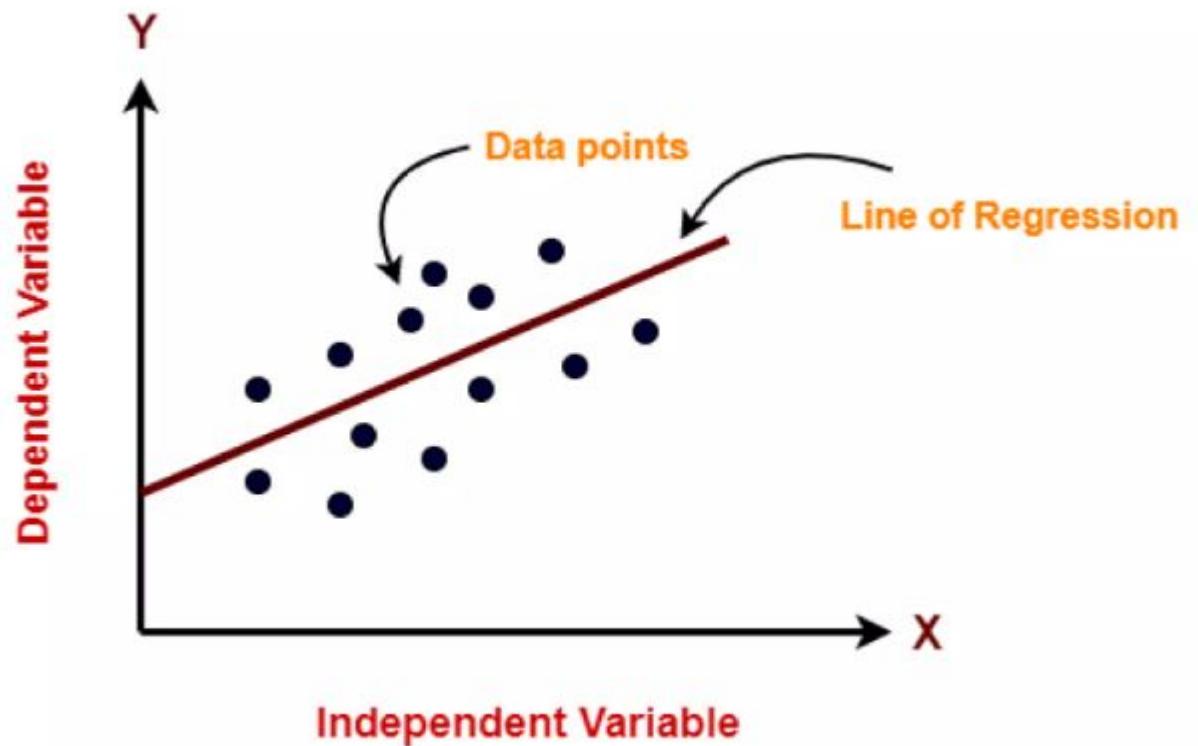
Based on the number of variables, there are two types of linear regression

- Simple linear regression
- Multiple linear regression

Linear Regression - Representation

Linear regression model represents the linear relationship between x & y via a sloped straight line.

The sloped straight line that fits the given data best is called as regression line or best fit line.

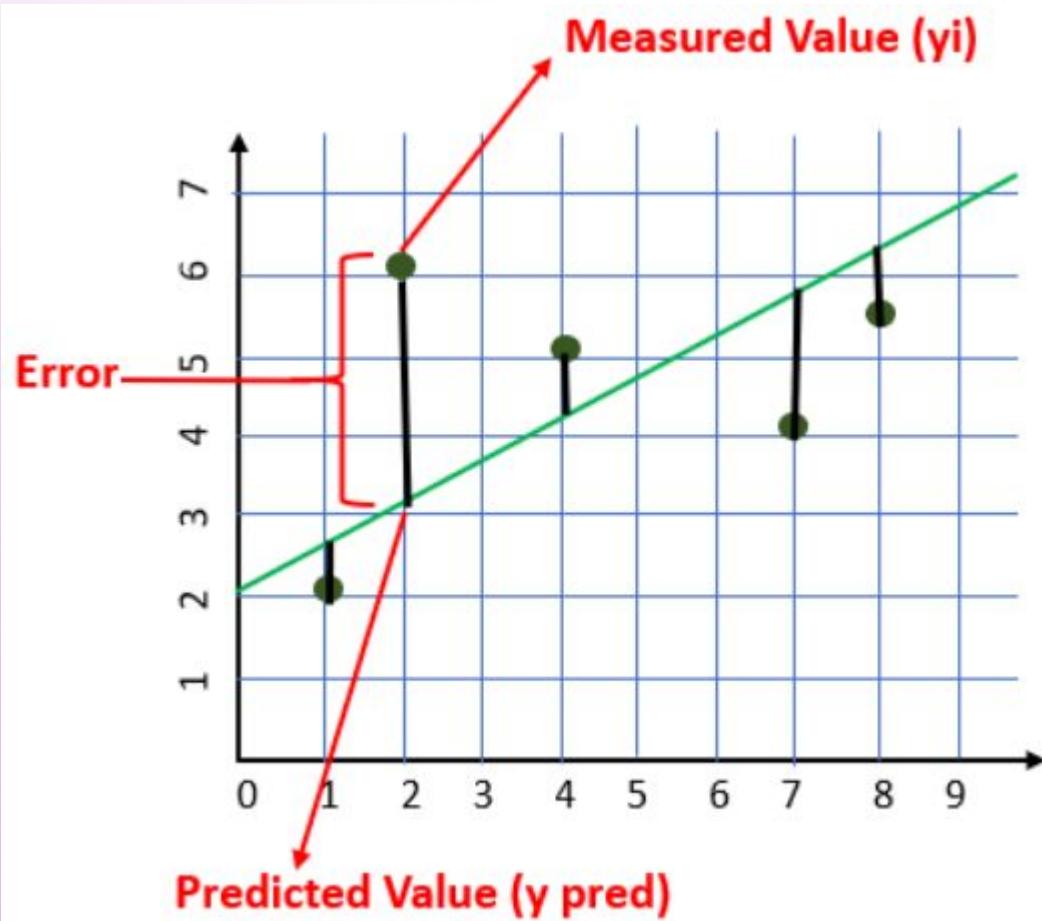


$$y = mx + c$$

$$\hat{y} = b_0 + b_1x$$

\hat{y} = predicted value of the dependent variable

Linear Regression - Cost Function

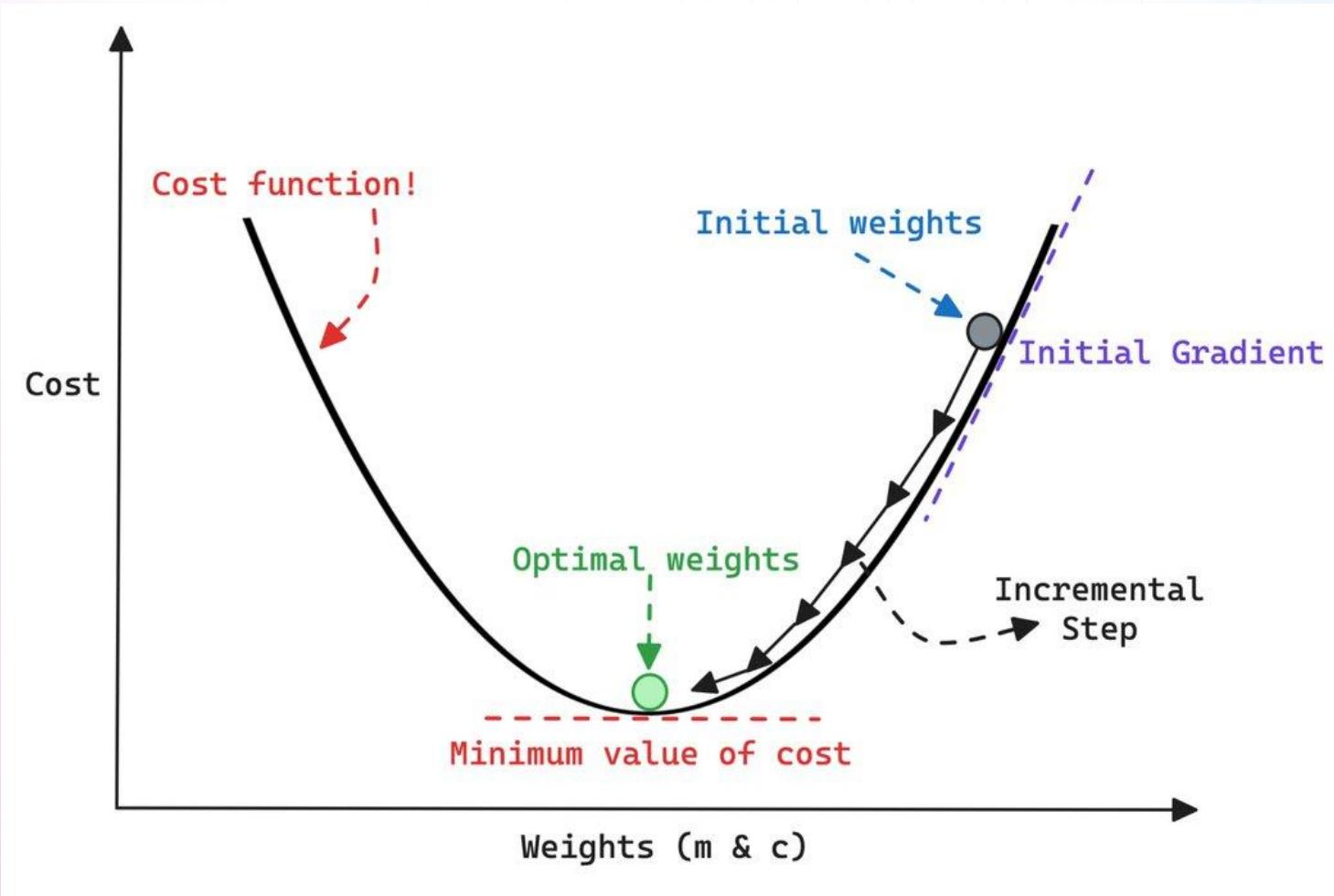


$$\text{Cost Function(MSE)} = \frac{1}{n} \sum_{i=0}^n (y_i - y_{i\ pred})^2$$

Replace $y_{i\ pred}$ with $mx_i + c$

$$\text{Cost Function(MSE)} = \frac{1}{n} \sum_{i=0}^n (y_i - (mx_i + c))^2$$

Linear Regression - Gradient Descent Algorithm



Linear Regression Code



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Generate new synthetic data for house sizes (in square feet) and prices (in $1000)
np.random.seed(42) # Changed seed for different data generation
sizes = np.random.rand(100, 1) * 3000 + 1000 # Sizes between 1000 and 4000 sqft
prices = 50 + 0.2 * sizes + np.random.randn(100, 1) * 20 # Prices in $1000 with some noise

# Fit the linear regression model
model = LinearRegression()
model.fit(sizes, prices)

# Generate predictions for the range of sizes
sizes_new = np.linspace(1000, 4000, 100).reshape(-1, 1)
prices_predict = model.predict(sizes_new)

# Plot the data and the regression line
plt.figure(figsize=(10, 6))
plt.scatter(sizes, prices, color="blue", label="Data points")
plt.plot(sizes_new, prices_predict, color="red", label="Regression line")
plt.xlabel("Size (sqft)")
plt.ylabel("Price (thousands of dollars)")
plt.title("Linear Regression: Predicting House Prices Based on Size")
plt.legend()
plt.grid(True)
plt.show()
```

House Price Prediction

Polynomial Regression

Simple
Linear
Regression

$$y = b_0 + b_1 x_1$$

Multiple
Linear
Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

Polynomial
Linear
Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$$

Polynomial Regression

In polynomial regression, the relationship between the independent variable x and the dependent variable y is modeled as a degree n polynomial

$$y = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n + \epsilon$$

Here:

- y is the dependent variable (the variable you are trying to predict).
- x is the independent variable (the input variable).
- b_0, b_1, \dots, b_n are the coefficients that need to be determined.
- n is the degree of the polynomial.
- ϵ is the error term, which captures the difference between the observed and predicted values.

Polynomial Regression Contd..

Overfitting: A high-degree polynomial might fit the training data very well but could lead to overfitting, where the model captures noise in the data rather than the underlying pattern. Overfitting can result in poor generalization to new data. It's important to choose an appropriate degree that balances complexity with the ability to generalize.

Applications:

- Modeling the trajectory of projectiles.
- Predicting growth rates in biology.
- Analyzing trends in economics.

Polynomial Regression Code - Car price prediction



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline

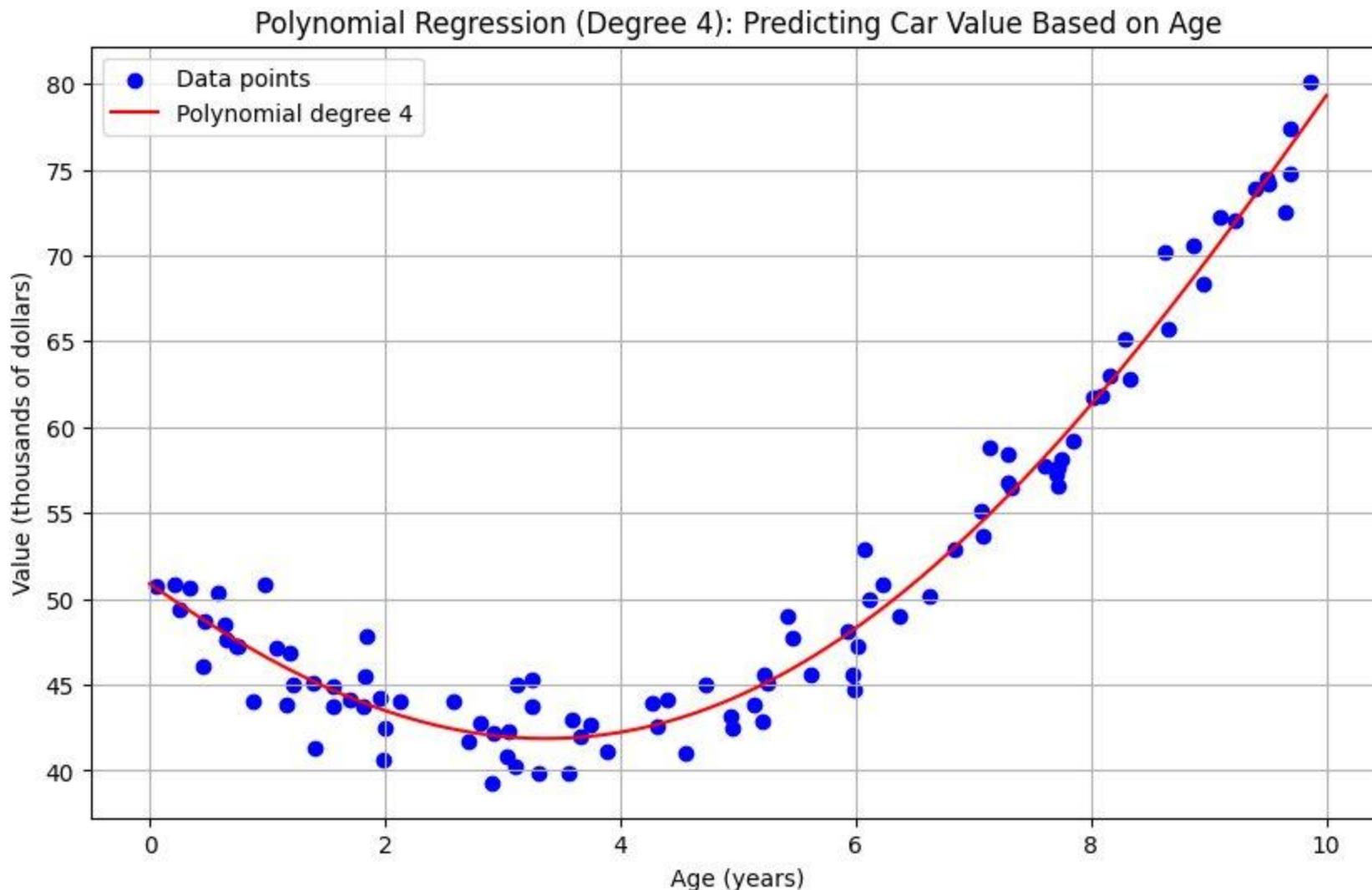
# Generate synthetic data
np.random.seed(42)
ages = np.random.rand(100, 1) * 10 # Car ages between 0 and 10 years
values = 50 - 5 * ages + 0.8 * ages**2 + np.random.randn(100, 1) * 2 # Non-linear relationship

# Polynomial feature transformation using sklearn
degree = 4 # Higher degree polynomial
poly_reg_model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
poly_reg_model.fit(ages, values)

# Generate predictions
ages_new = np.linspace(0, 10, 100).reshape(-1, 1)
values_predict = poly_reg_model.predict(ages_new)

# Plot the data and the polynomial regression curve
plt.figure(figsize=(10, 6))
plt.scatter(ages, values, color="blue", label="Data points")
plt.plot(ages_new, values_predict, color="red", label=f"Polynomial degree {degree}")
plt.xlabel("Age (years)")
plt.ylabel("Value (thousands of dollars)")
plt.title("Polynomial Regression (Degree 4): Predicting Car Value Based on Age")
plt.legend()
plt.grid(True)
plt.show()
```

Polynomial Regression Code Output



k-Nearest Neighbour Classification (kNN)

The KNN algorithm works based on the idea that similar things are closer to each other.

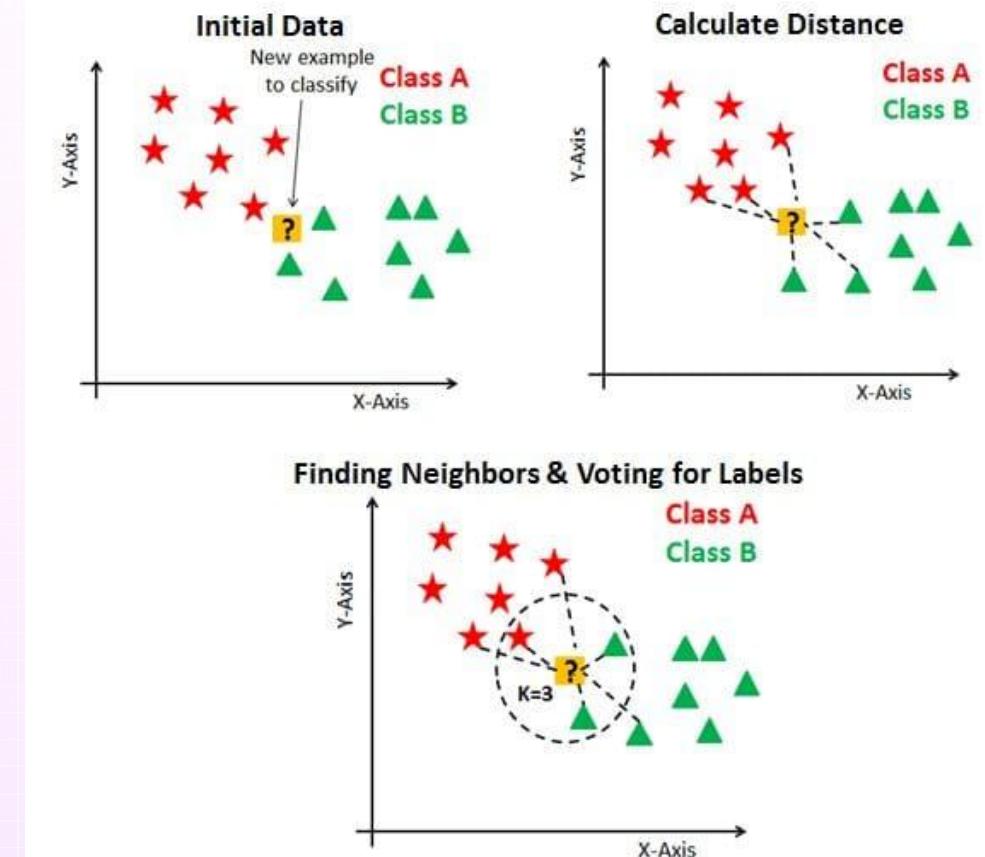
Unlike the previous models, kNN does not build model from the training data

To classify a test instance d , define k -neighbourhood P as k nearest neighbours of d

Count number n of training instances in P that belong to class C_j

Estimate $\Pr(C_j | d)$ as n/k

No training is needed. Classification time is linear to training set size for each test case



k-Nearest Neighbour Classification (kNN)

Algorithm kNN(D , d , k):

1. Compute the distance between d and every example in D
2. Choose the k examples in D that are nearest to d , denote the set by P
3. Assign d the class that is the most frequent class in P (or the majority class)

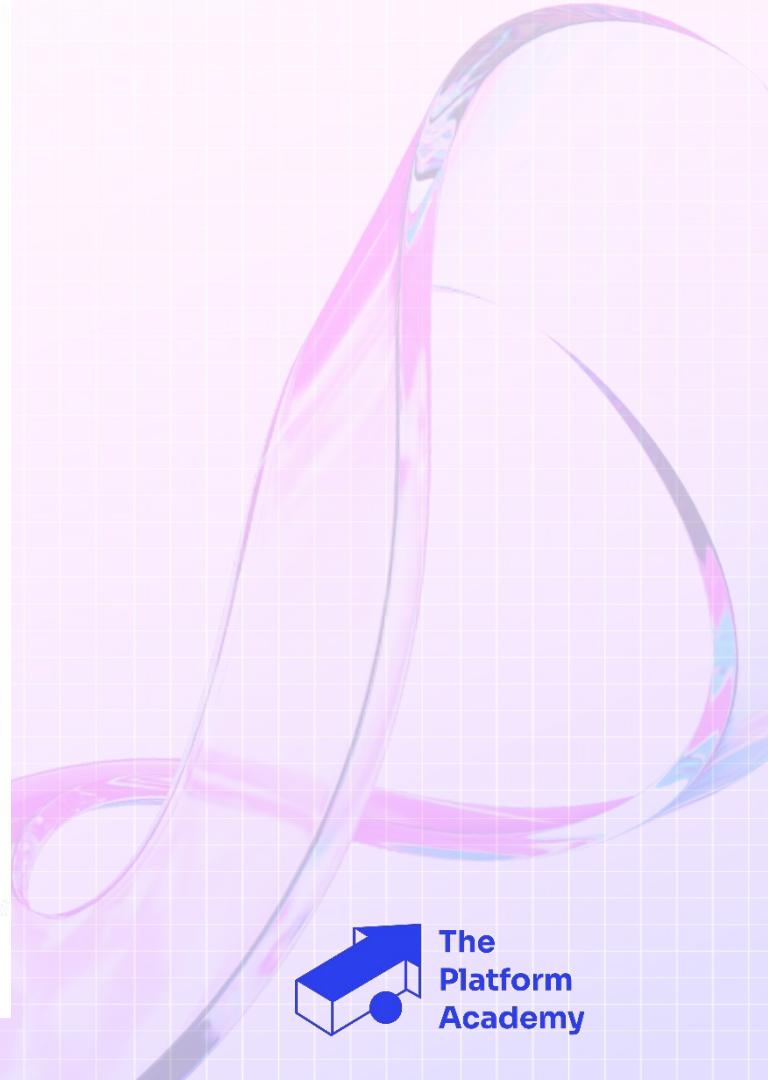
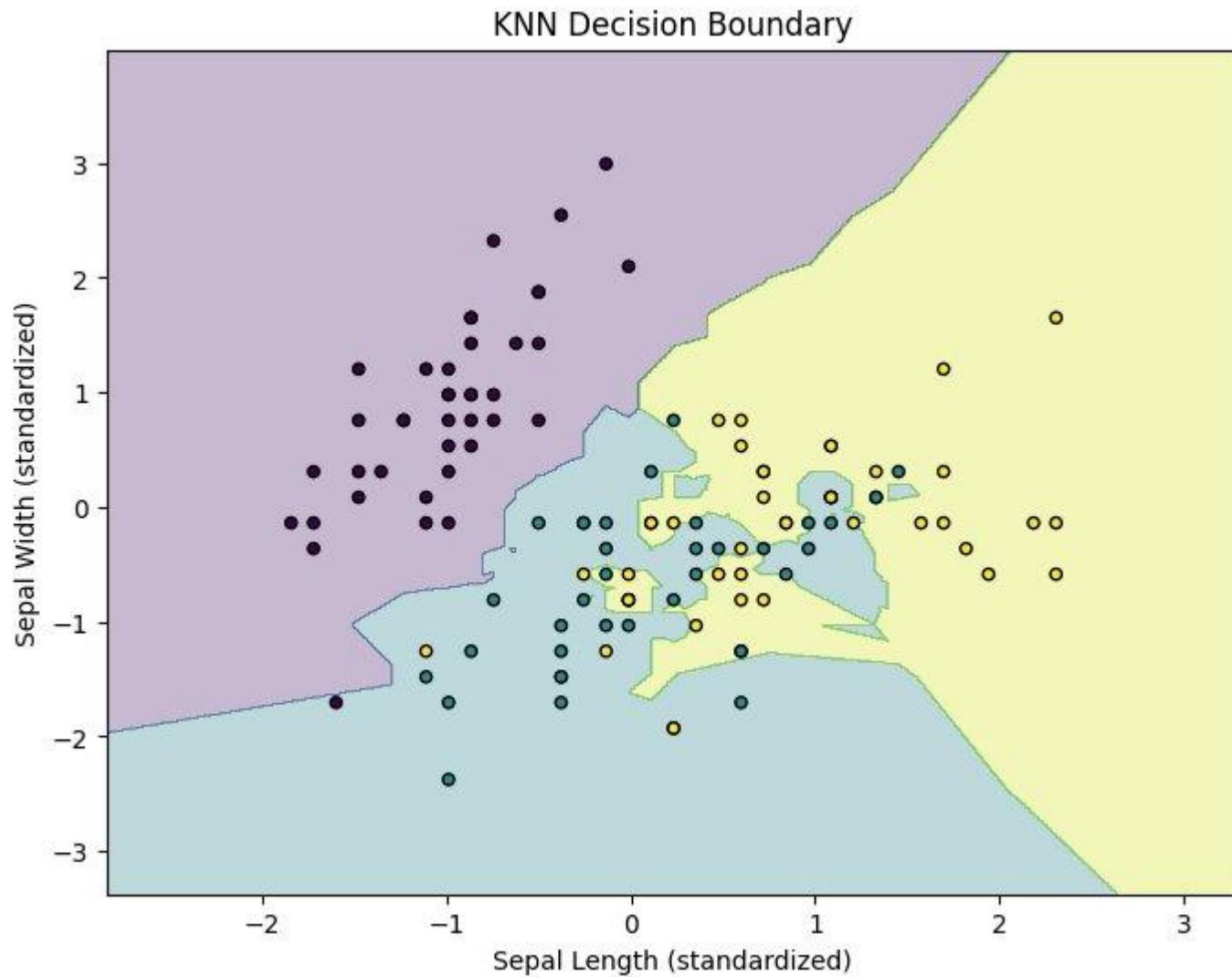
k is usually chosen empirically via a validation set or cross-validation by trying a range of k values

Accuracy is used as the metric for evaluating K performance

Applications:

- Image Recognition
- Medical Diagnosis
- Recommendation Systems

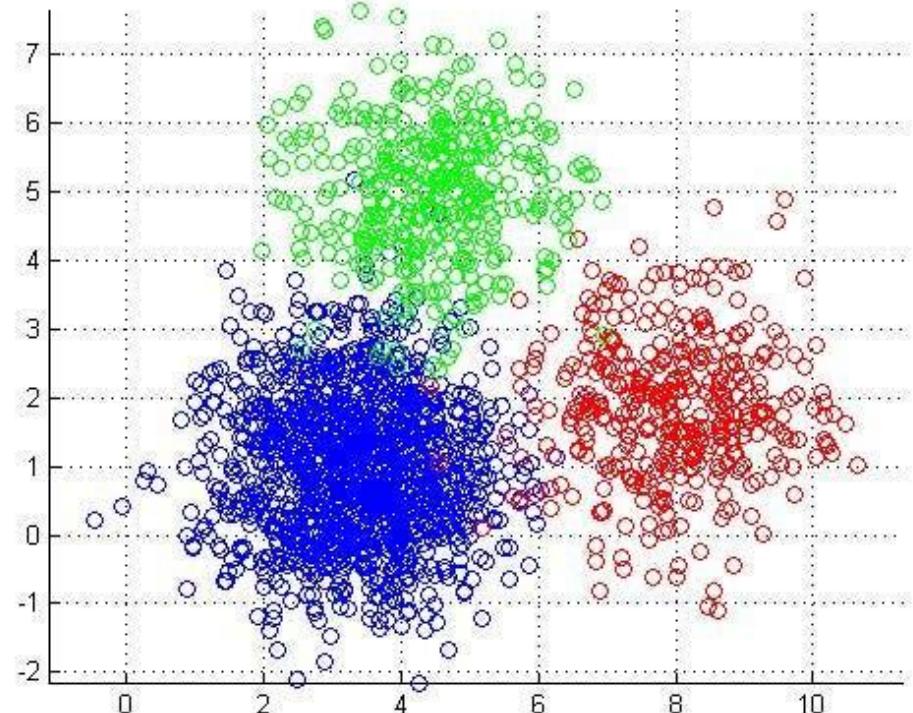
k-Nearest Neighbour Classification (kNN) - Code Output



Clustering

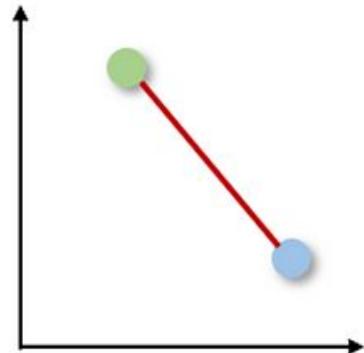
Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

Clustering is the classification of objects into different groups, or more precisely, the partitions of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

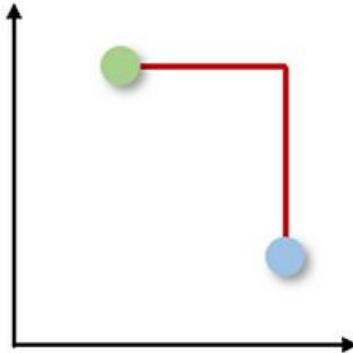


Common Distance Measures

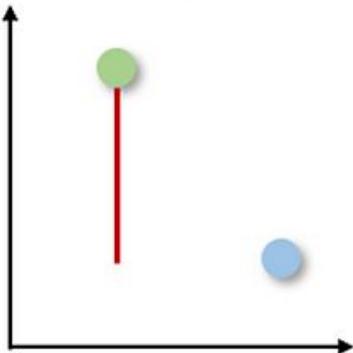
Euclidean



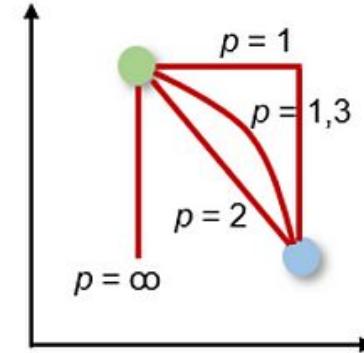
Manhattan



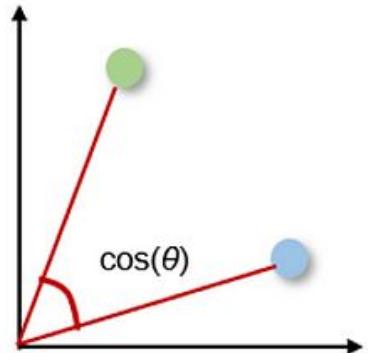
Chebyshev



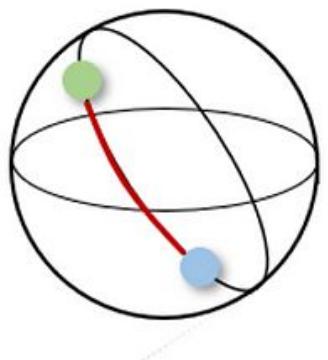
Minkowski



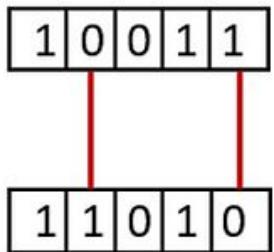
Cosine



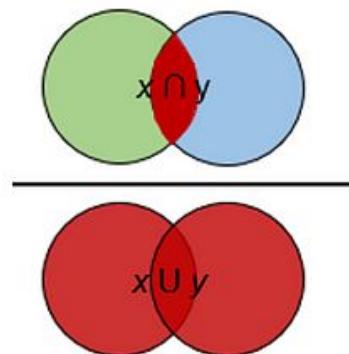
Haversine



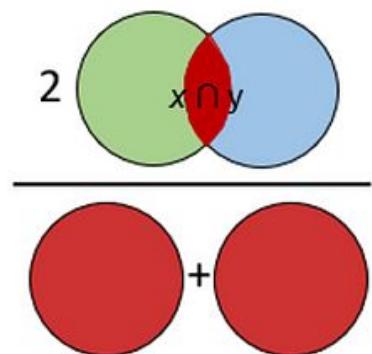
Hamming



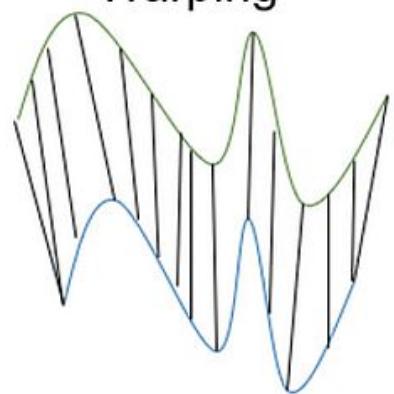
Jaccard



Sörensen-Dice



Dynamic Time Warping



K Means Clustering

Input: D is a dataset containing n objects, k is the number of clusters

Output: A set of k clusters

Steps:

1. Randomly choose k objects form D as the initial cluster centroids
2. For each of the objects in D do
 - a. Compute distance between the current objects and k cluster centroids
 - b. Assign the current object to that cluster to which it is closest
3. Compute the cluster centers of each cluster. These become the new cluster centroids.
4. Repeat step 2-3 until the convergence criterion is satisfied
5. Stop

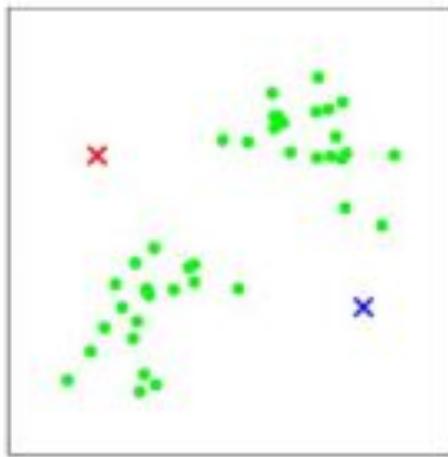
Convergence Criteria: Any one of the following are termination conditions for the algorithm

- Number of maximum iterations permissible
- No change of centroid values in any cluster
- Zero (or no significant) movements of objects from one cluster to another
- Cluster quality reaches to a certain level of acceptance

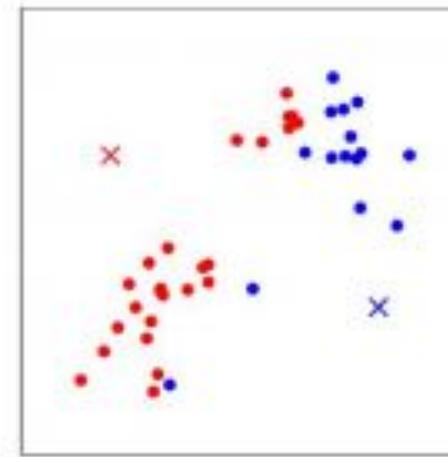
K Means Clustering



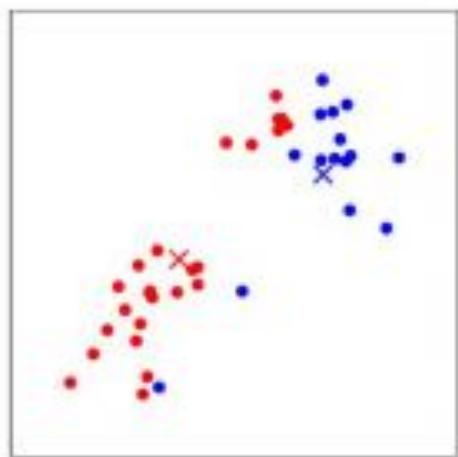
(a)



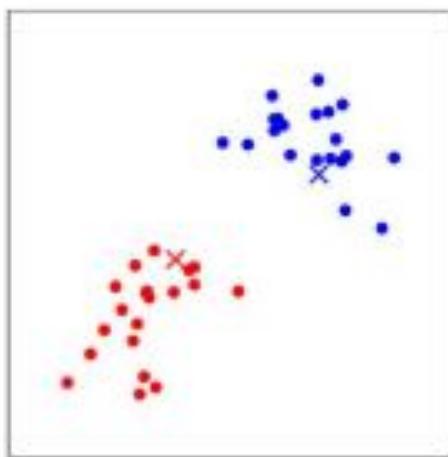
(b)



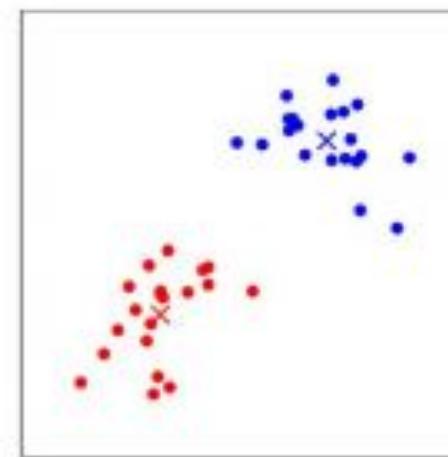
(c)



(d)



(e)

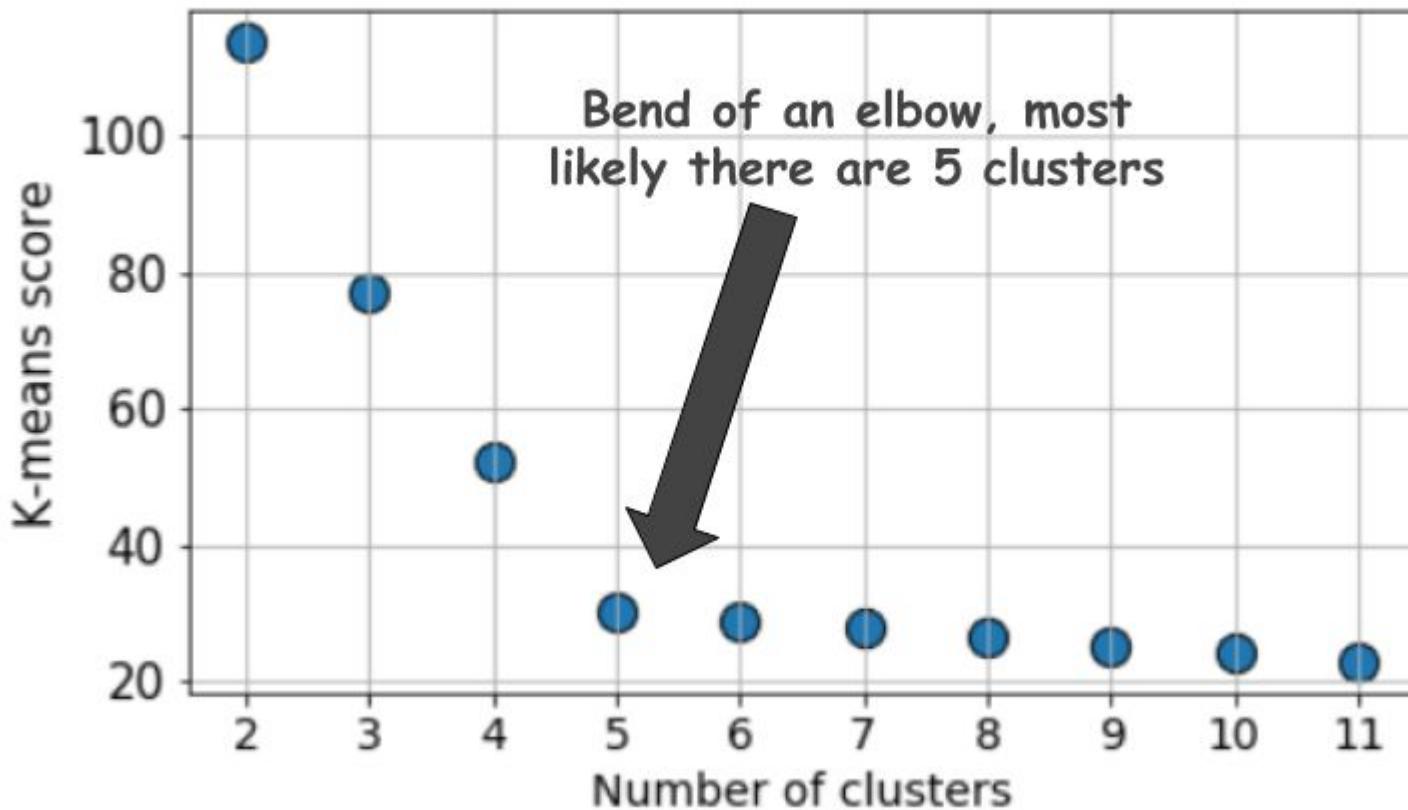


(f)

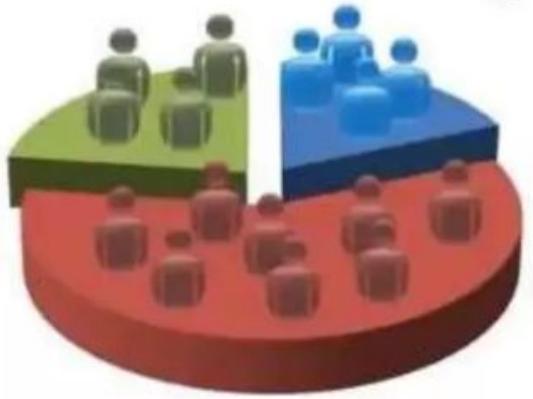


K Means Clustering - finding right k

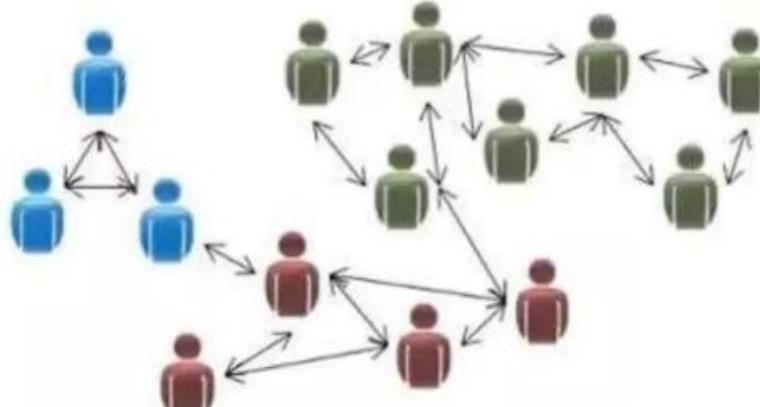
The elbow method for determining number of clusters



K Means Clustering Applications



Market segmentation



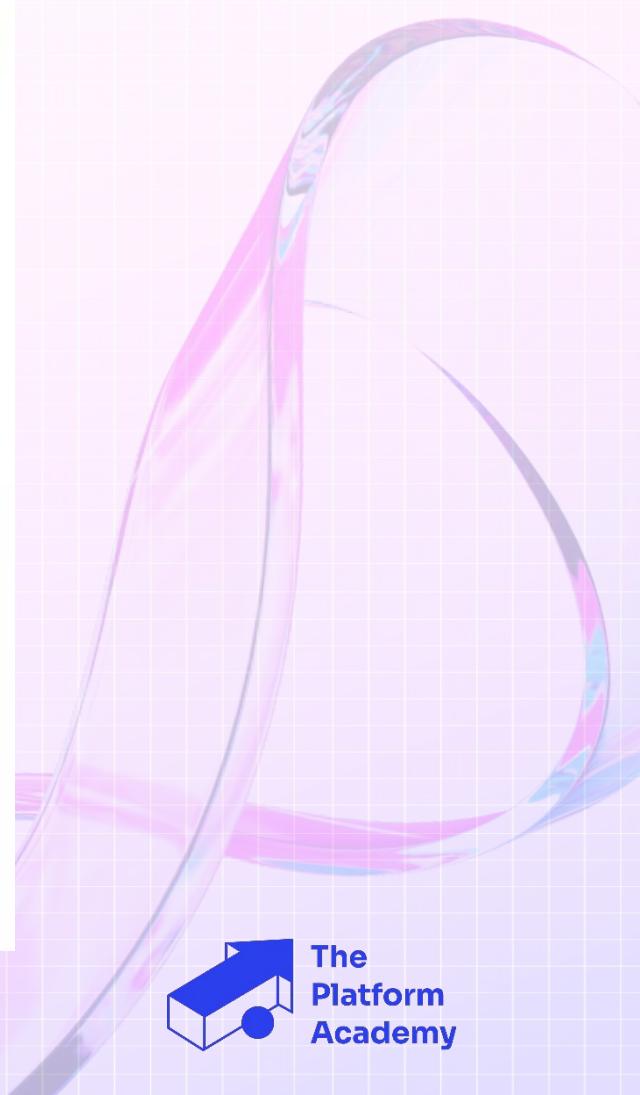
Social network analysis



Organize computing clusters



Astronomical data analysis



The
Platform
Academy

K Means Clustering Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# Generate synthetic data
np.random.seed(42)
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.6, random_state=42)

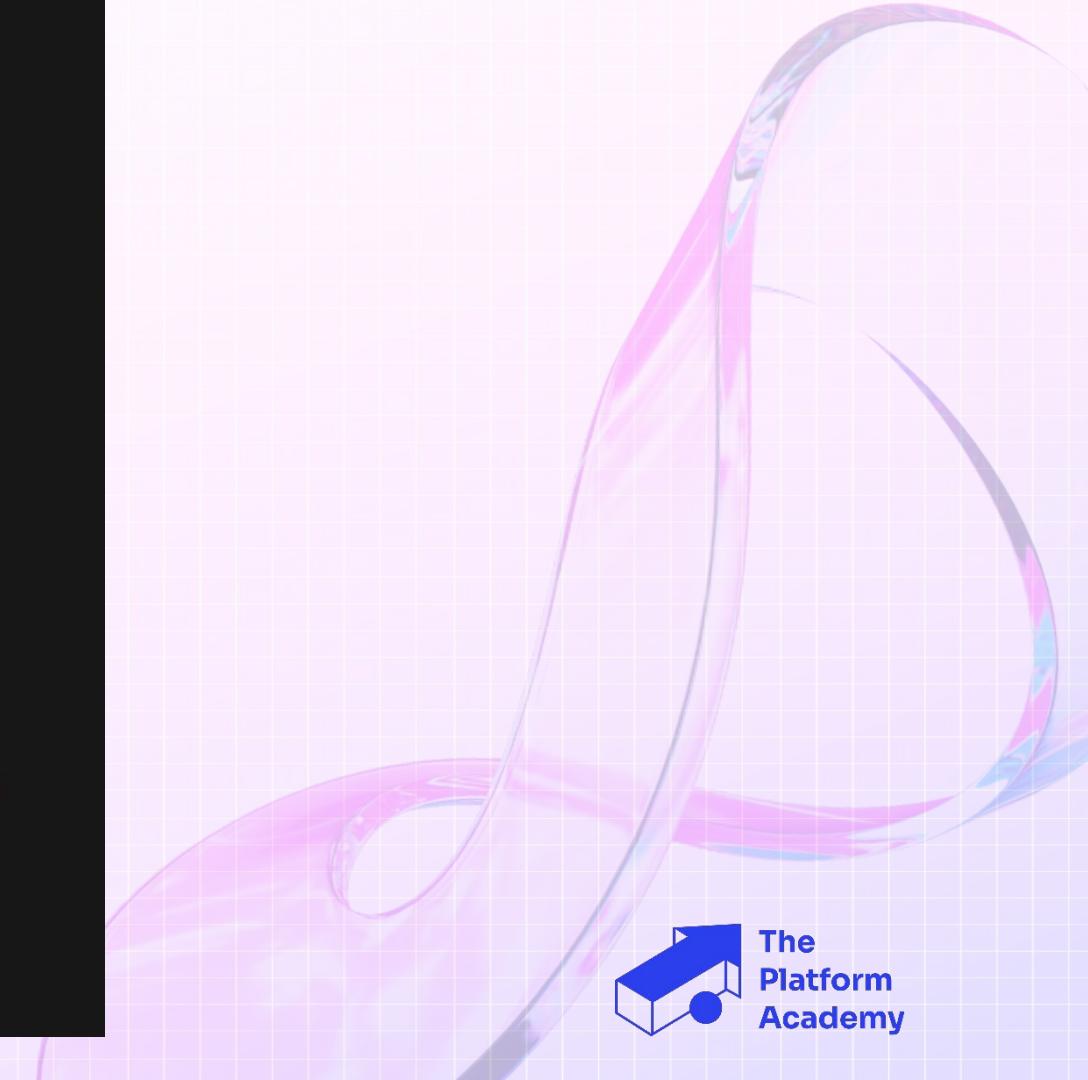
# Plot the original data
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.title("Original Data")
plt.show()

# Initialize the KMeans model
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(X)

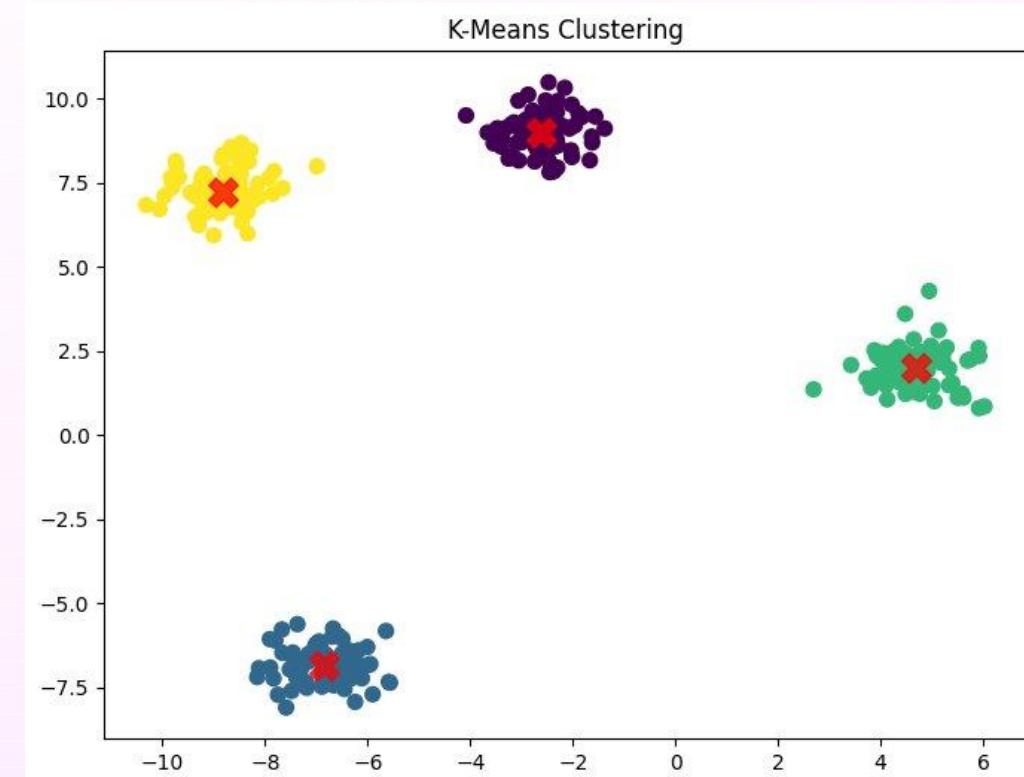
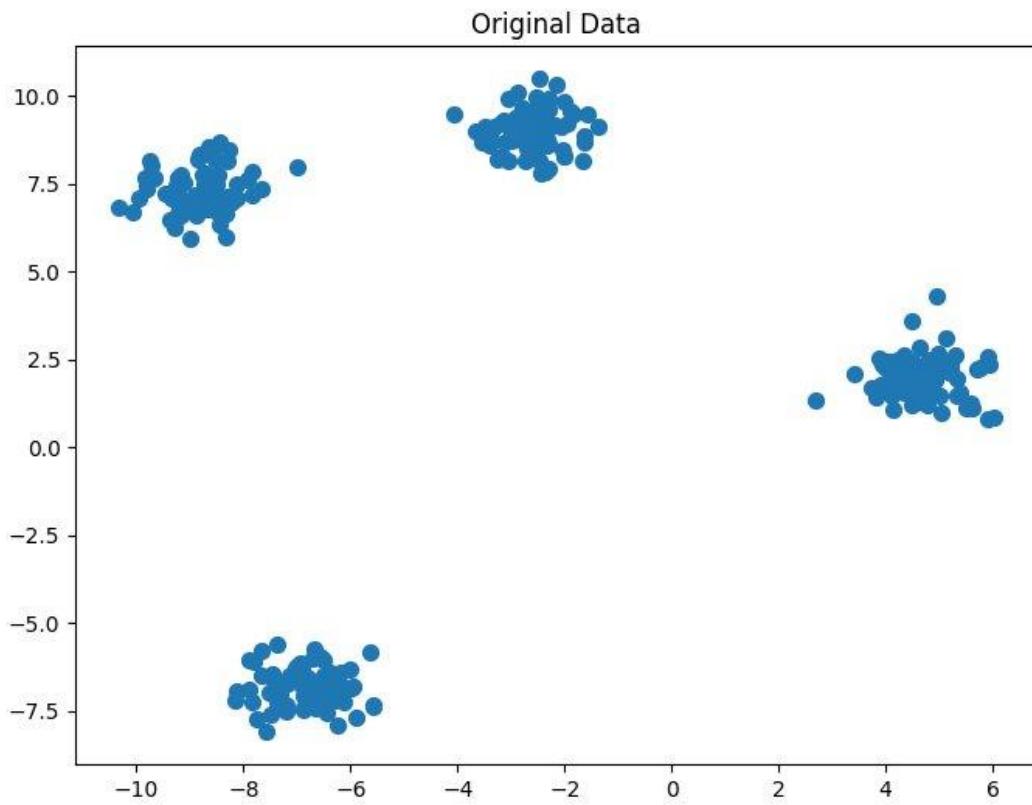
# Get the cluster centers and labels
centers = kmeans.cluster_centers_
labels = kmeans.labels_

# Plot the clustered data
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title("K-Means Clustering")
plt.show()

# Print the final cluster centers
print("Cluster Centers:\n", centers)
```



K Means Clustering - Code Output



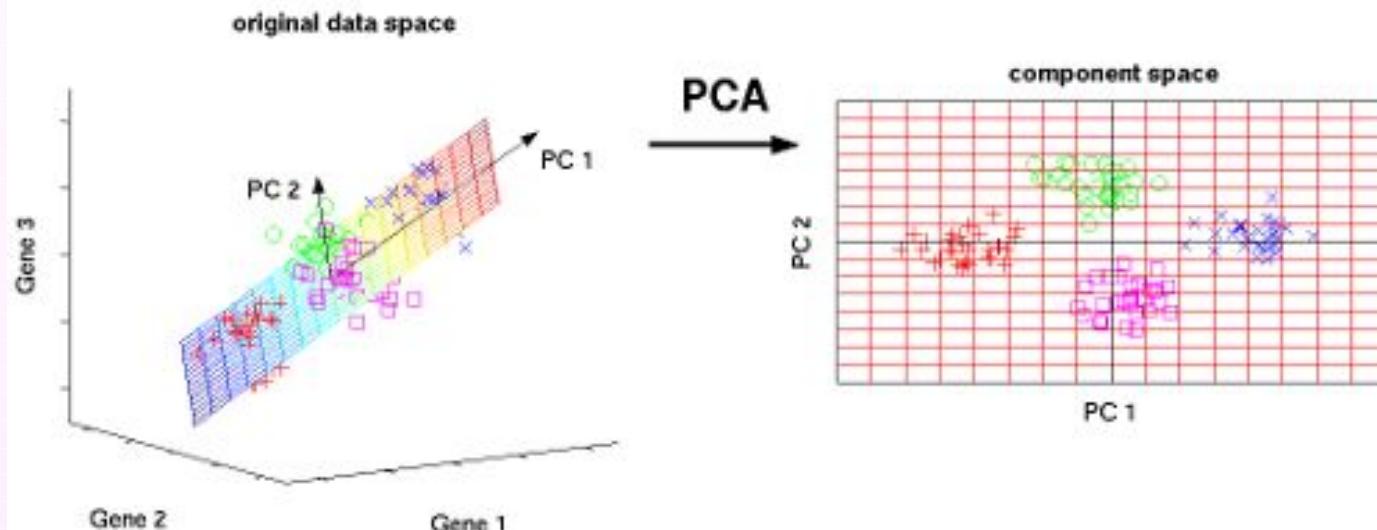
Principal Component Analysis (PCA)

PCA is used to reduce dimensions of data without much loss of information

PCA does an orthogonal linear transformation that transfers the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (first principal component), the second greatest variance lies on the second coordinate (second principal component), and so on.

Applications:

- Face recognition
- Image compression
- Gene expression analysis



Principal Component Analysis - Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
X = iris.data # Features (sepal length, sepal width, petal length, petal width)
y = iris.target # Target (class labels: Setosa, Versicolour, Virginica)

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

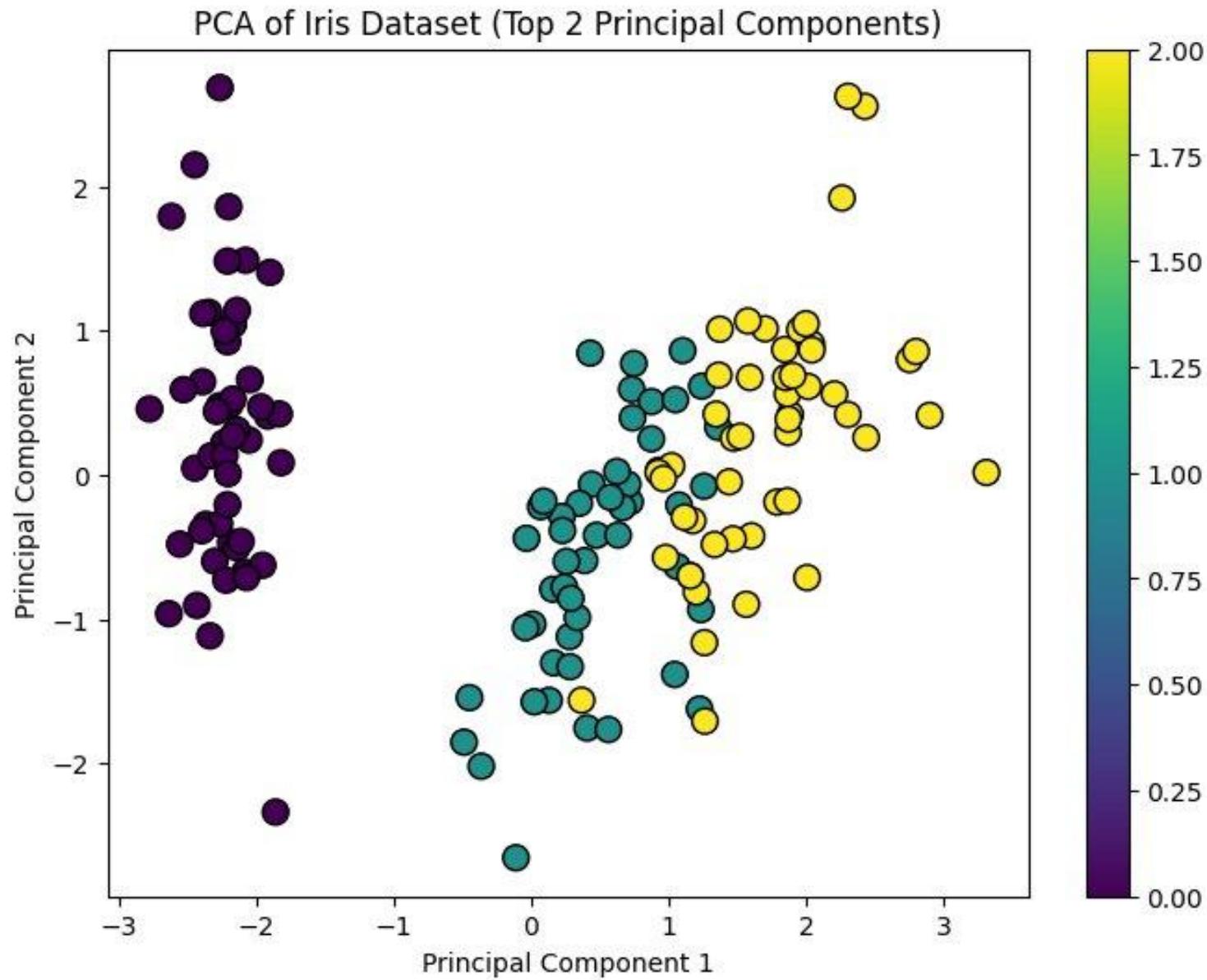
# Perform PCA to reduce to the top 3 principal components
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Print the explained variance ratio for the top 3 components
print("Explained variance ratio of top 2 principal components:", pca.explained_variance_ratio_)

# Print the top 2 principal components
print("\nTop 2 principal components (PCA components):\n", pca.components_)

# Visualizing the data in the first two principal components
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', edgecolor='k', s=100)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset (Top 2 Principal Components)')
plt.colorbar()
plt.show()
```

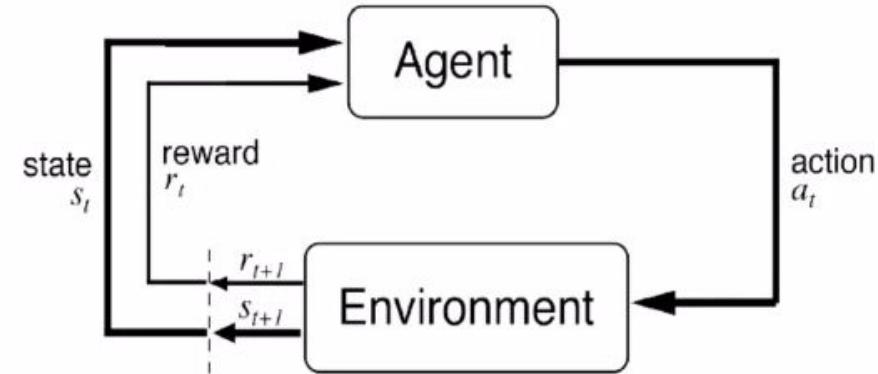
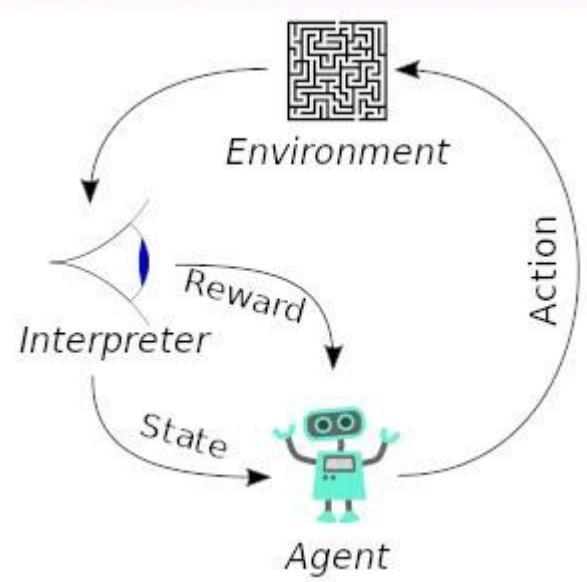
Principal Component Analysis - Code Output



Reinforcement Learning

Algorithm learns a policy of how to act in a given environment

Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm



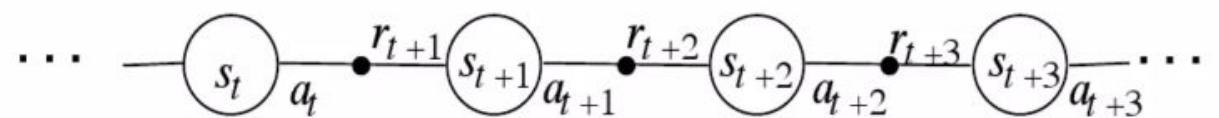
Agent and environment interact at discrete time steps : $t = 0, 1, 2, \dots$

Agent observes state at step t : $s_t \in S$

produces action at step t : $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \mathcal{R}$

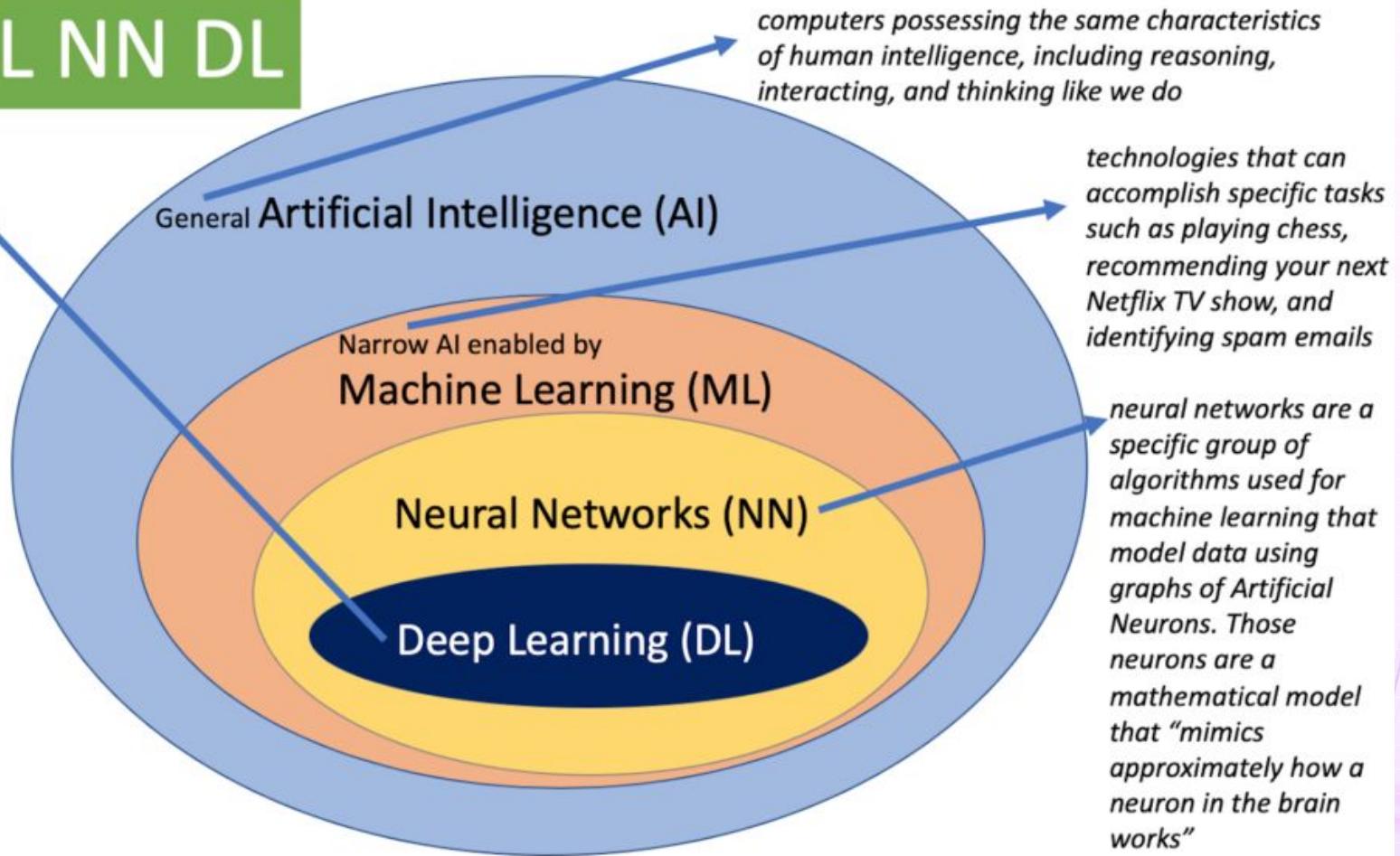
and resulting next state : s_{t+1}



Neural Networks

AI ML NN DL

the word "deep" comes from the fact that DL algorithms are trained/run on deep neural networks. These are just neural networks with (usually) three or more "hidden" layers



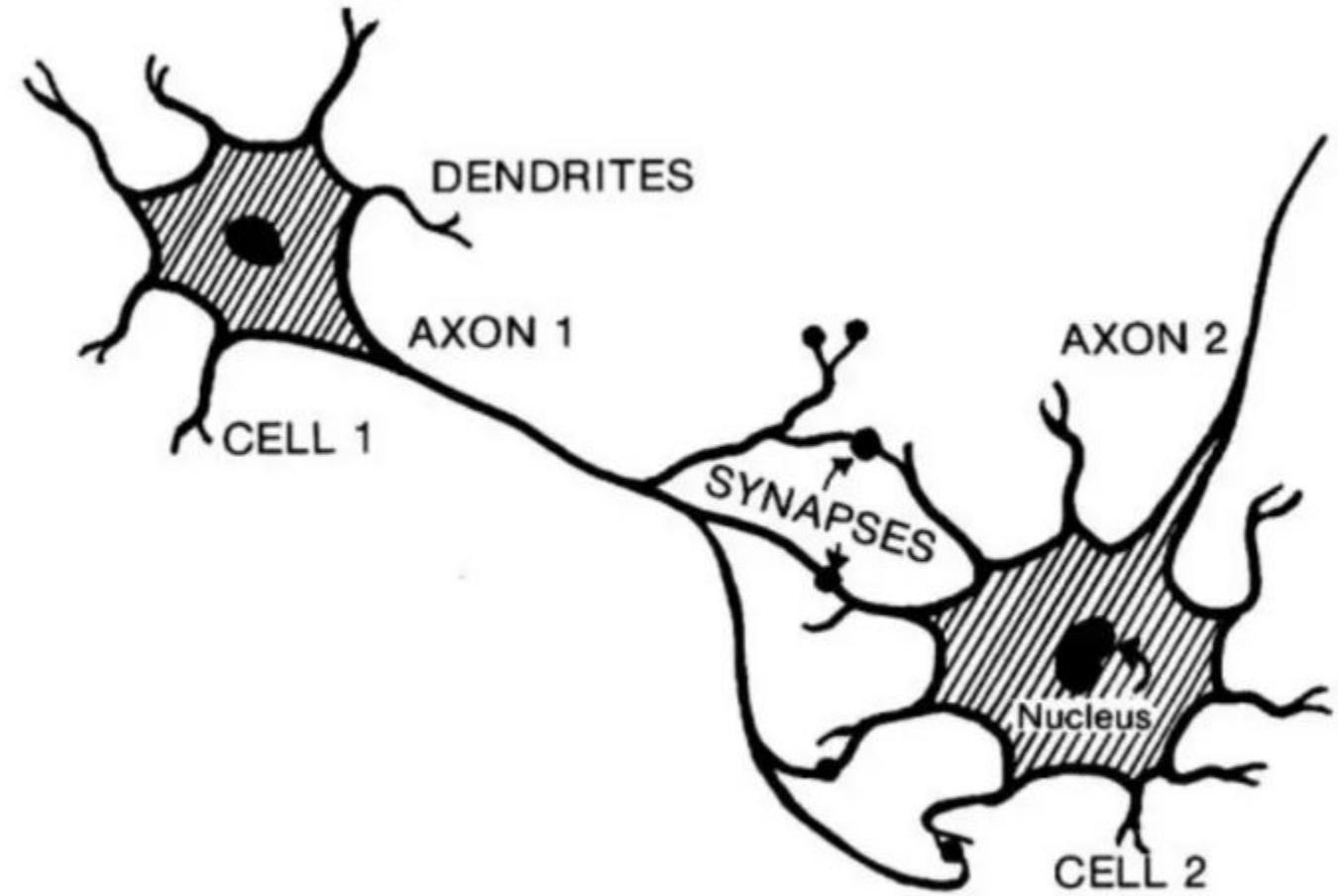
The
Platform
Academy

Neural Networks - Neuron

Brain has a huge number of neurons (10 billion neurons and about 60000 billions of interconnections)

Brain is a very complex, non-linear, parallel computer

- Dendrites accept inputs from other neurons
- Axon transmits impulses to other neurons
- Synapses pass electrical signals or impulses from one neuron to another



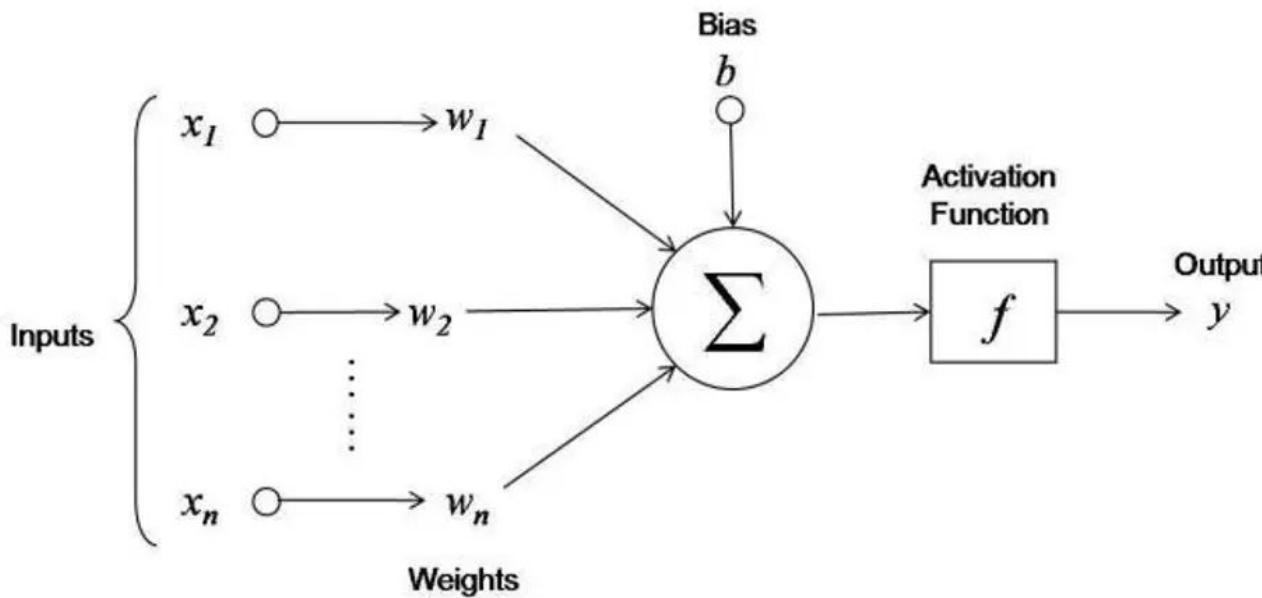
Artificial Neural Networks

Computing systems inspired by the biological neural networks

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain

Each connection, like the synapses in a biological brain, can transmit a signal to other neurons

Neural Networks - Neuron Model

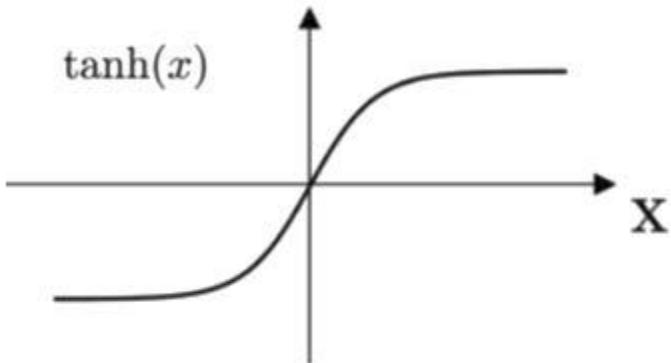


After taking a weighted sum of the inputs plus the bias ($W_1X_1 + W_2*X_2 + \dots + Wn*Xn + b$), we pass this value to some activation function f , which then gives us the output of the given neuron.

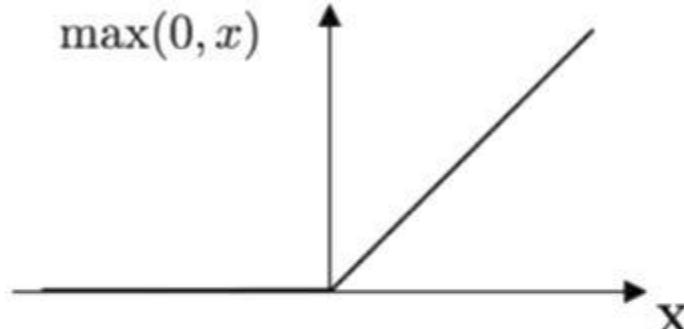
In this case, each of the X_i values is the output of a neuron from the previous layer, while W_i is our neuron's weights assigned to each input X_i .

Neural Networks - Activation Functions

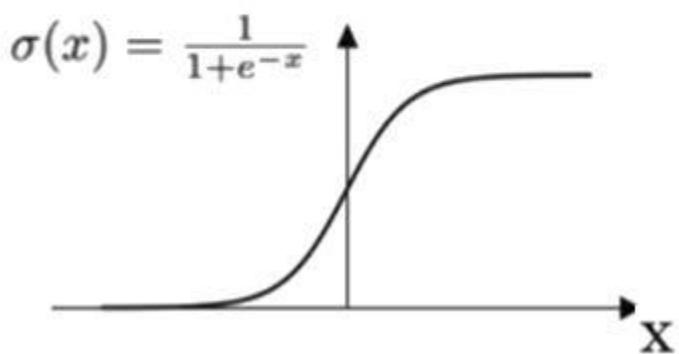
Tanh



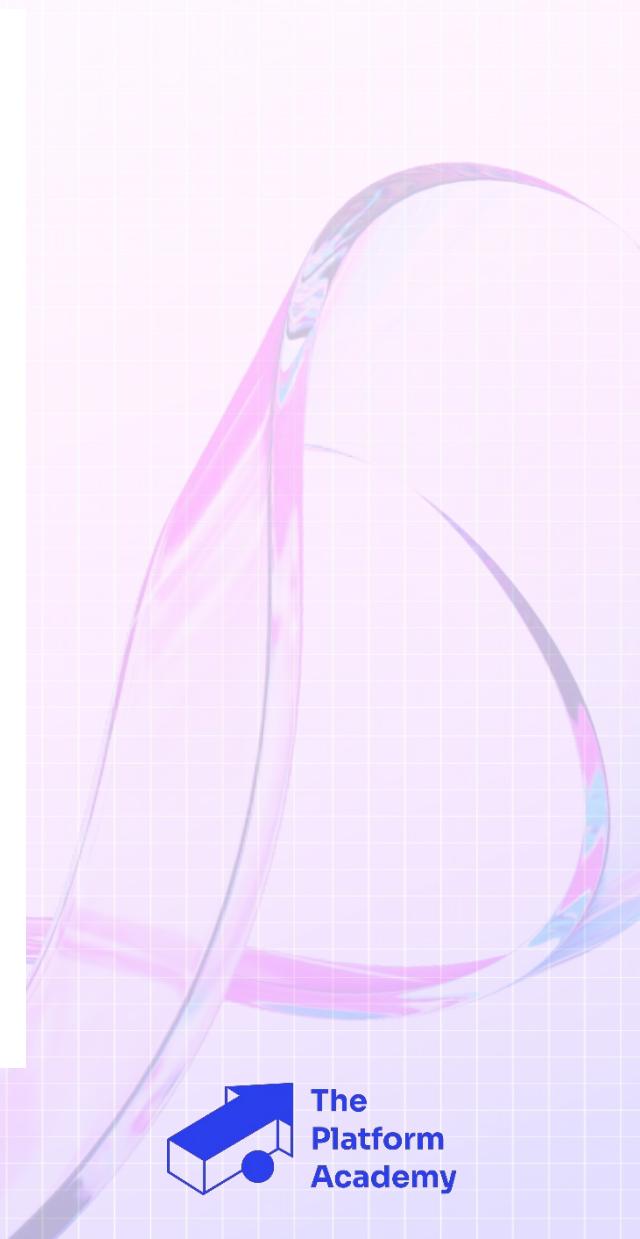
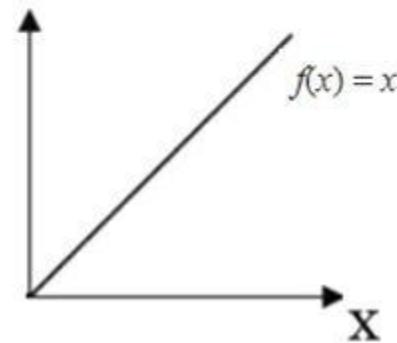
ReLU



Sigmoid



Linear



The
Platform
Academy

Neural Networks

A Simple Neural Network

Use recent three days' average temperature to predict tomorrow's average temperature.

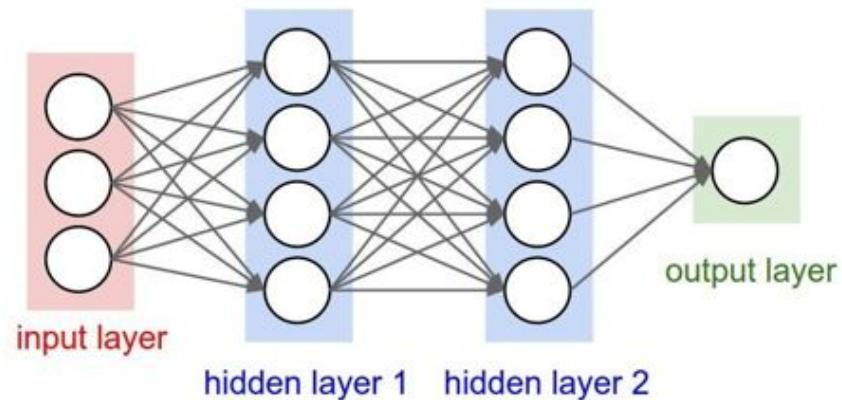
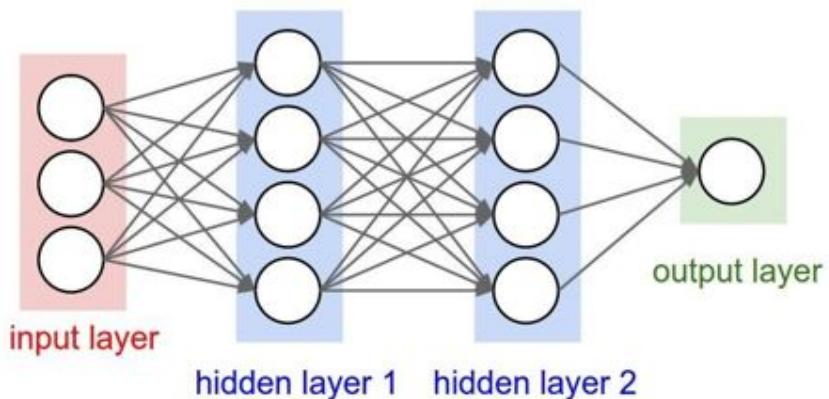


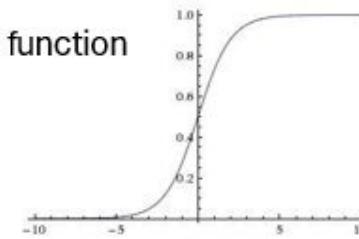
Image from CS231N

Neural Networks

A Simple Neural Network



Sigmoid function



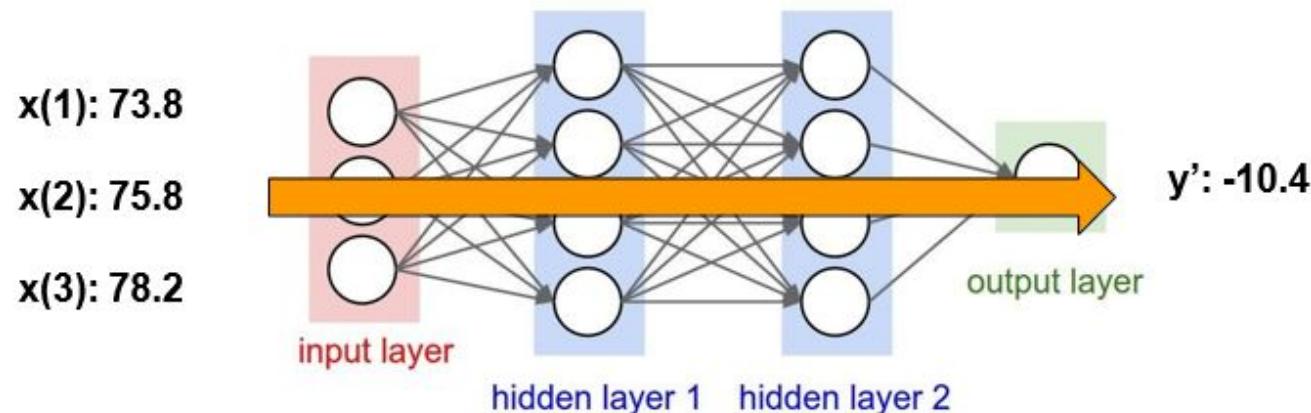
W1, b1, W2, b2, W3, b3
are network parameters
that need to be learned.

```
# forward-pass of a 3-layer neural network:  
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)  
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)  
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)  
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)  
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

From CS231N

Neural Networks

Neural Network: Forward Pass

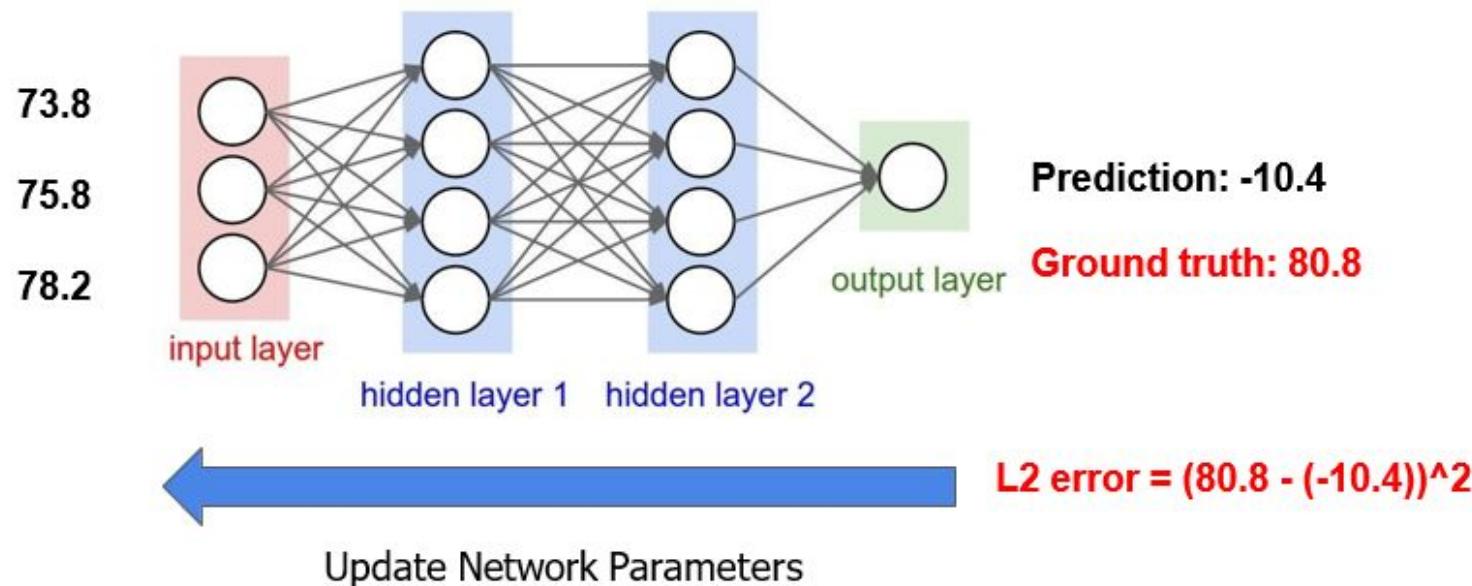


$$y' = W_3 f(W_2 f(W_1 x + b_1) + b_2) + b_3$$

From CS231N

Neural Networks

Neural Network: Backward Pass

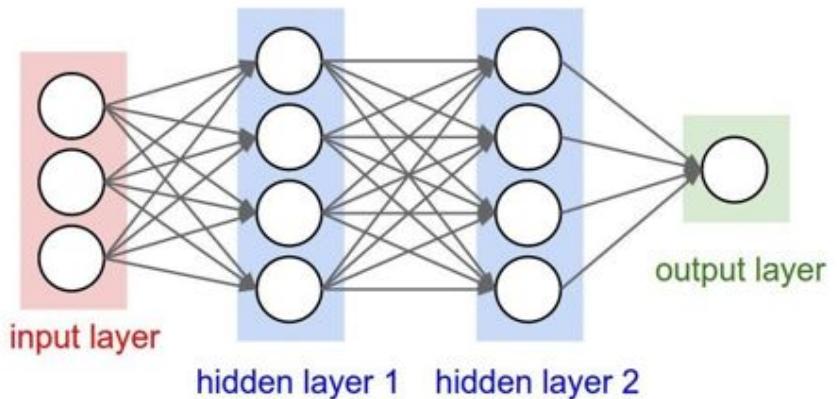


$$\text{Minimize: } L(x, y; W, b) = \sum_{i=1}^N (W_3 f(W_2 f(W_1 x_i + b_1) + b_2) + b_3) - y_i)^2$$

Given N training pairs: $\{x_i, y_i\}_{i=1}^N$

Neural Networks

A Simple Neural Network



Model: Multi-Layer Perceptron (MLP) $y' = W_3f(W_2f(W_1x + b_1) + b_2) + b_3$

Loss function: L2 loss $l(y, y') = (y - y')^2$

Optimization: Gradient descent $W = W - \eta \frac{\partial L}{\partial W}$



Neural Networks

Epoch 000,000 Learning rate 0.03 Activation Tanh Regularization None Regularization rate 0 Problem type Classification

DATA Which dataset do you want to use?

FEATURES Which properties do you want to feed in?

2 HIDDEN LAYERS

OUTPUT Test loss 0.504 Training loss 0.503

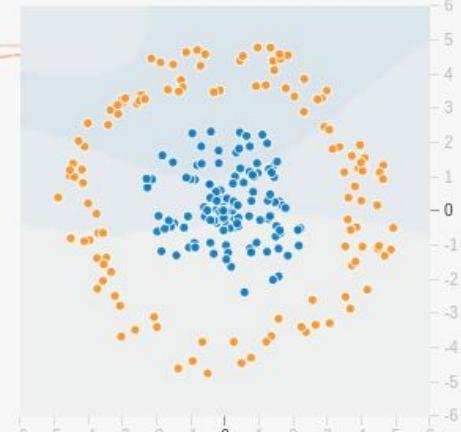
Ratio of training to test data: 50% Noise: 0 Batch size: 10

REGENERATE

4 neurons
2 neurons

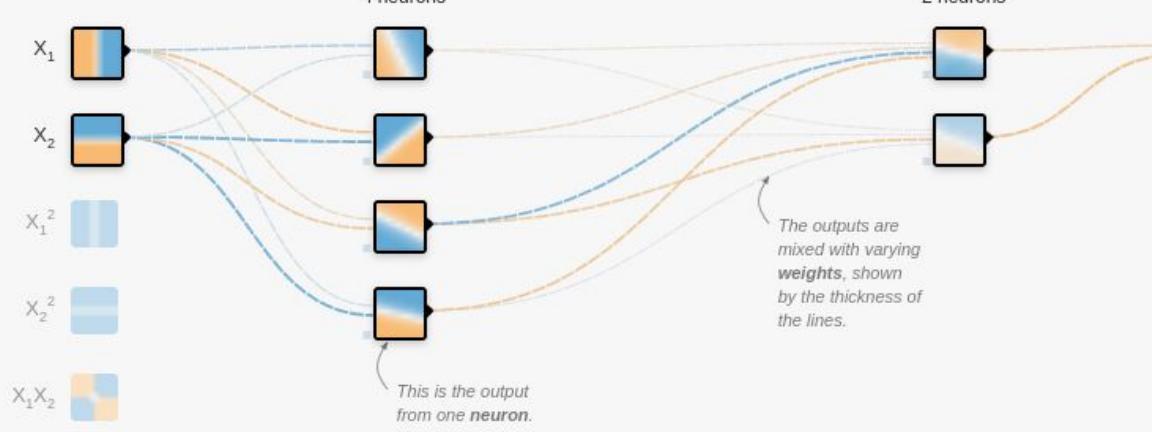
This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.



Colors show data, neuron and weight values. -1 0 1

Show test data Discretize output



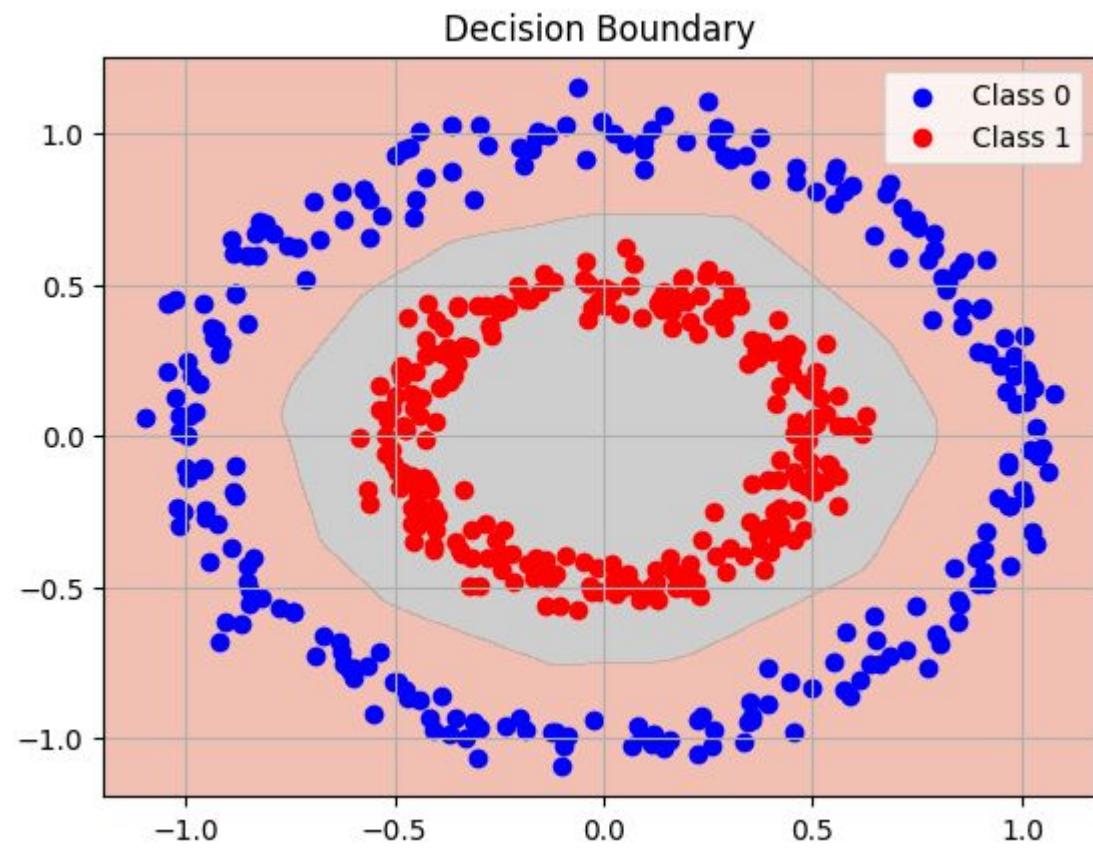
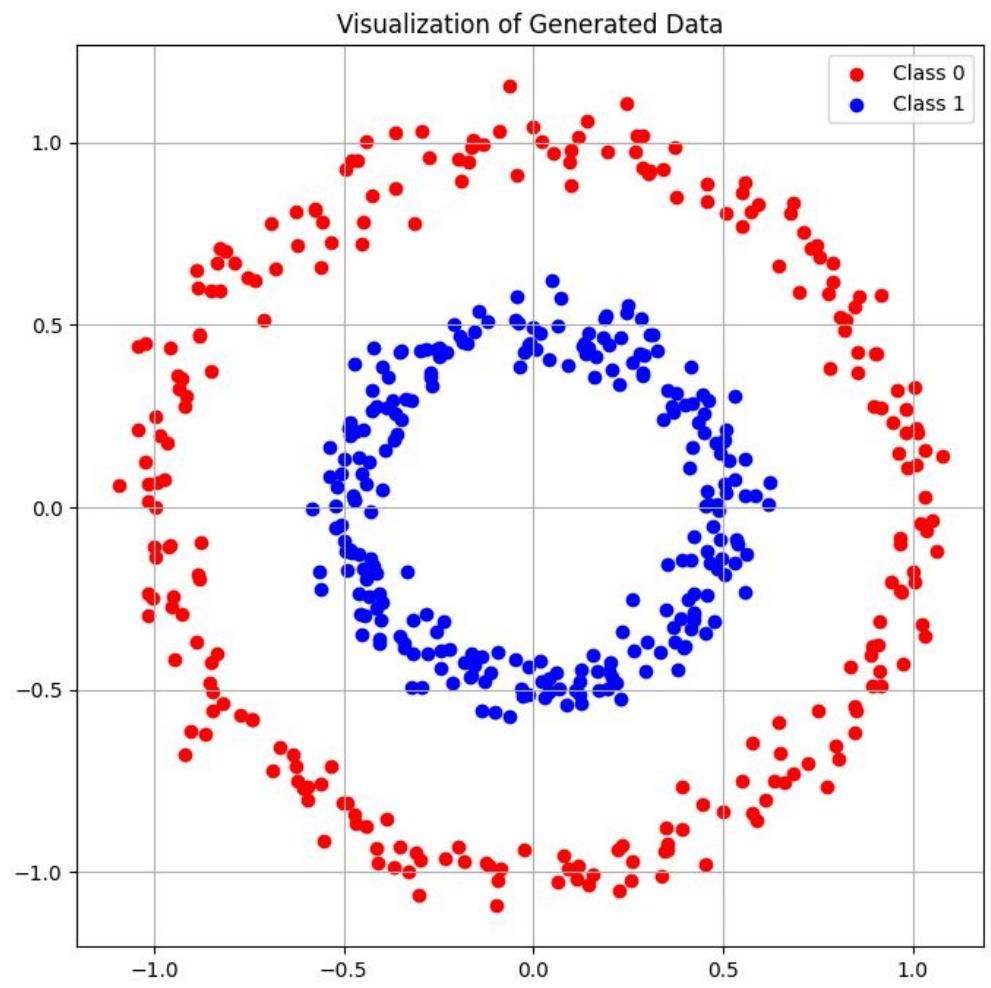
A neural network diagram with two hidden layers. The first layer has 4 neurons and the second has 2. Inputs X1 and X2 feed into the first layer. The outputs of the first layer feed into the second. The final output is a classification. The diagram shows the flow of data through the neurons and the resulting decision boundary on a 2D plot.

<https://playground.tensorflow.org/>

Neural Networks - Code

Open Neural_Networks.ipynb

Neural Networks - Code Output



Convolutional Neural Network (CNN)

A CNN is a type of deep learning algorithm primarily used for processing and analyzing visual data, such as images and videos.

CNNs are particularly effective for tasks like image classification, object detection, and image segmentation due to their ability to automatically and adaptively learn **spatial hierarchies** of features from input images.

Traditional neural networks struggle with image data due to the high dimensionality and spatial structure. CNNs address these challenges by leveraging the spatial hierarchies in images, making them effective for tasks like image recognition.

Convolutional Neural Network (CNN)



The
Platform
Academy

Convolutional Neural Network (CNN)

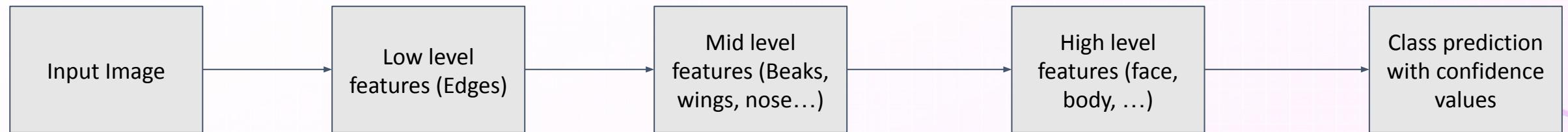
Round, elongated head with orange or black beak

Long white neck, square shape



Oval-shaped white body with or without large white symmetric blobs (wings)

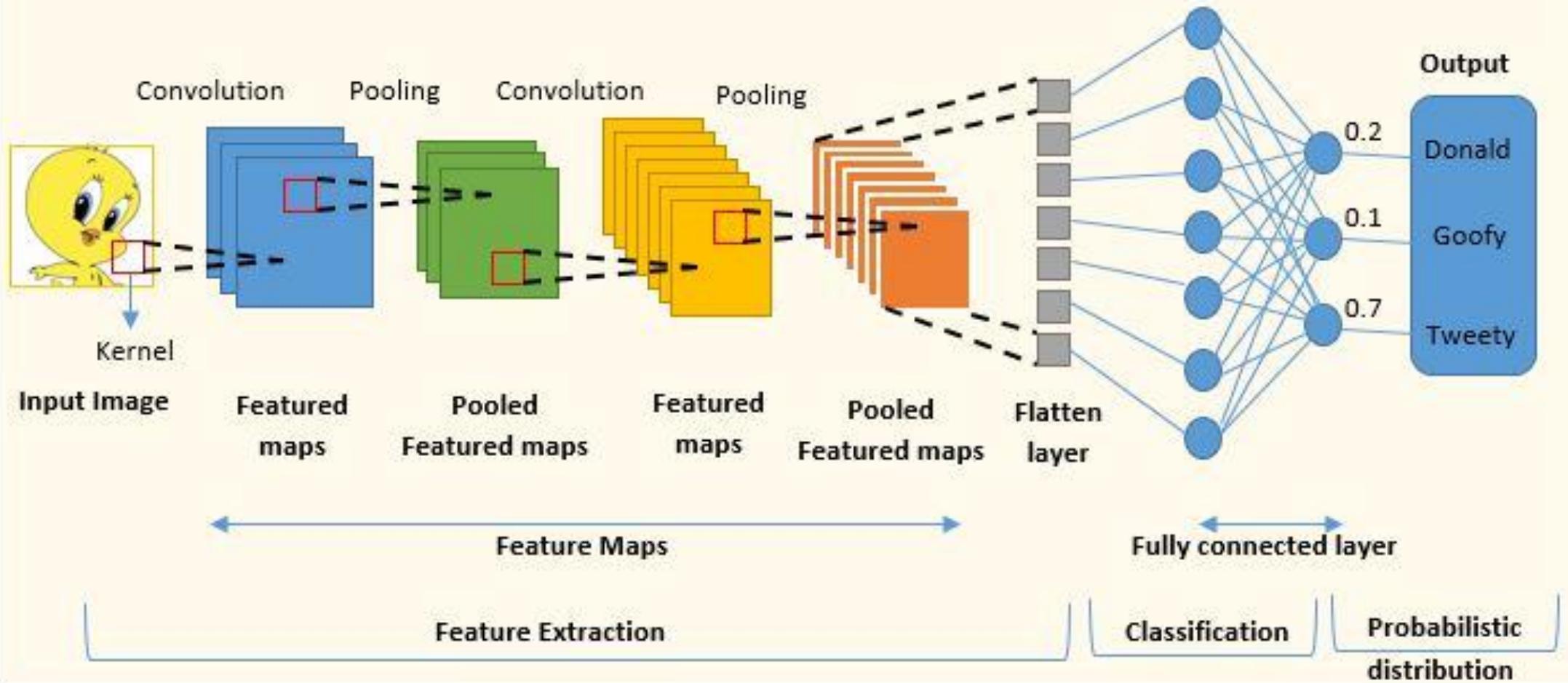
Convolutional Neural Network (CNN)



CNN learns the “features” by itself !!

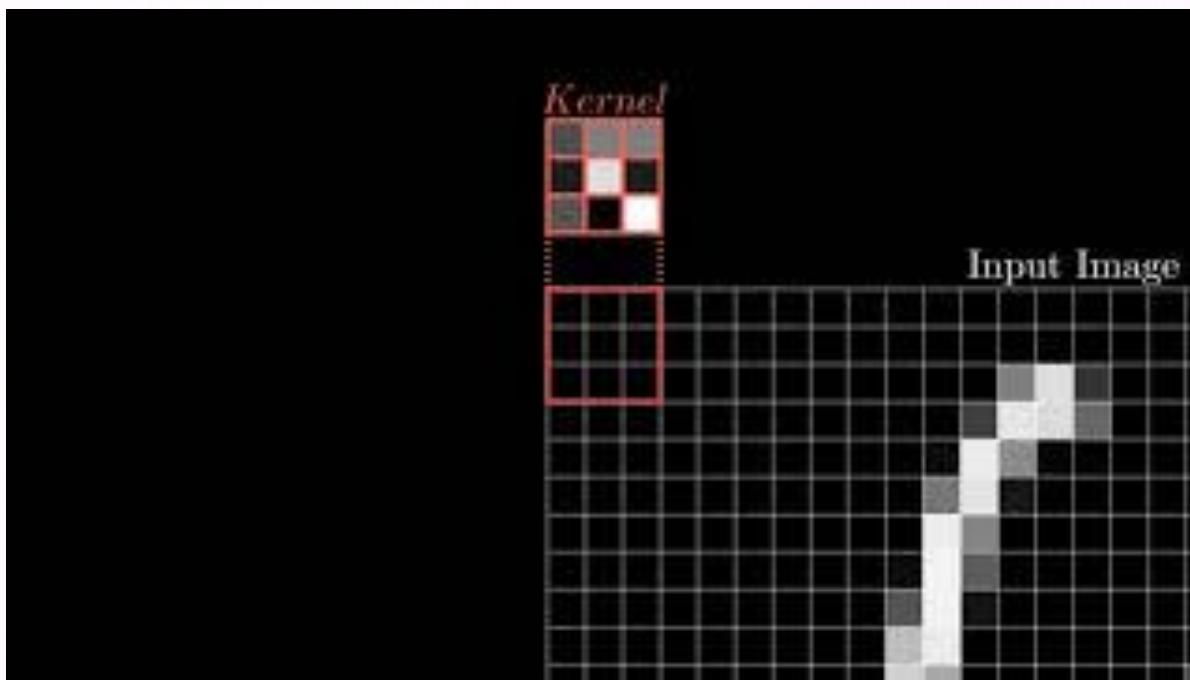
Convolutional Neural Network (CNN)

A Typical Convolutional Neural Network (CNN)

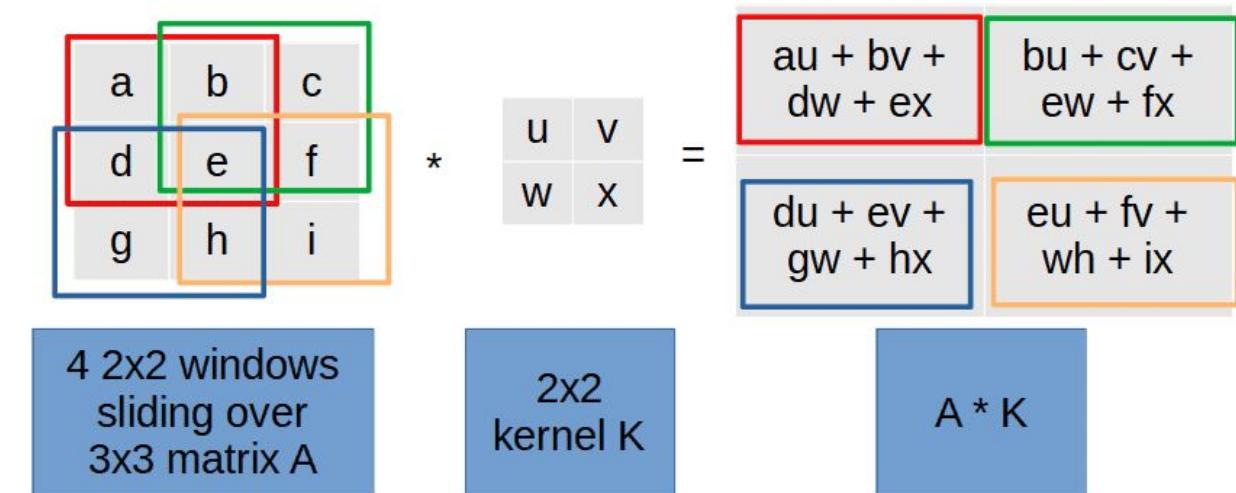


CNN - What is Convolution?

A convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another function f . It therefore "blends" one function with another.

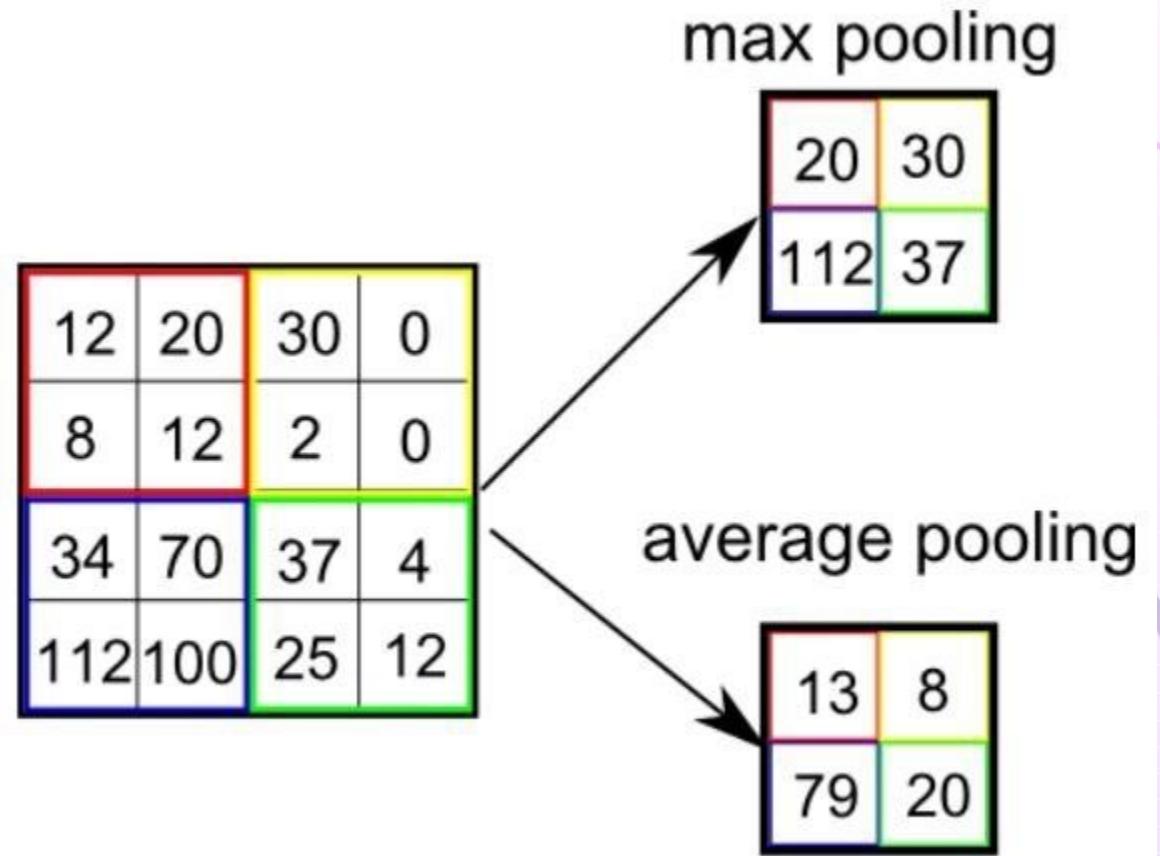


$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$



Convolutional Neural Networks - Pooling

Pooling layer (usually inserted in between conv layers) is used to reduce spatial size of the input, thus reduce number of parameters and overfitting



CNN - Sequence of layers

We have an input image using multiple filters to create various feature maps.

Each feature map of size (C, C) is pooled to generate a $(C/2, C/2)$ output (for a standard 2×2 pooling)

The above pattern is referred to as one Convolutional Neural Network layer or one unit. Multiple such CNN layers are stacked on top of each other to create deep Convolutional Neural Network networks.

The output of the convolution layer contains features, and these features are fed into a dense neural network.

CNN - Image Kernels Explained Visually

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

outline ▾



An **outline** kernel (also called an "edge" kernel) is used to highlight large differences in pixel values. A pixel next to neighbor pixels with close to the same intensity will appear black in the new image while one next to neighbor pixels that differ strongly will appear white.

<https://setosa.io/ev/image-kernels/>

Convolutional Neural Networks - Code

Open CNN.ipynb

Large Language Models

An LLM is a type of AI that is trained on a massive dataset of text. This allows the model to learn the statistical relationships between words and phrases, which in turn allows it to generate text, translate languages, write different kinds of creative content and answer your questions in an informative way

Here are some common LLMs:

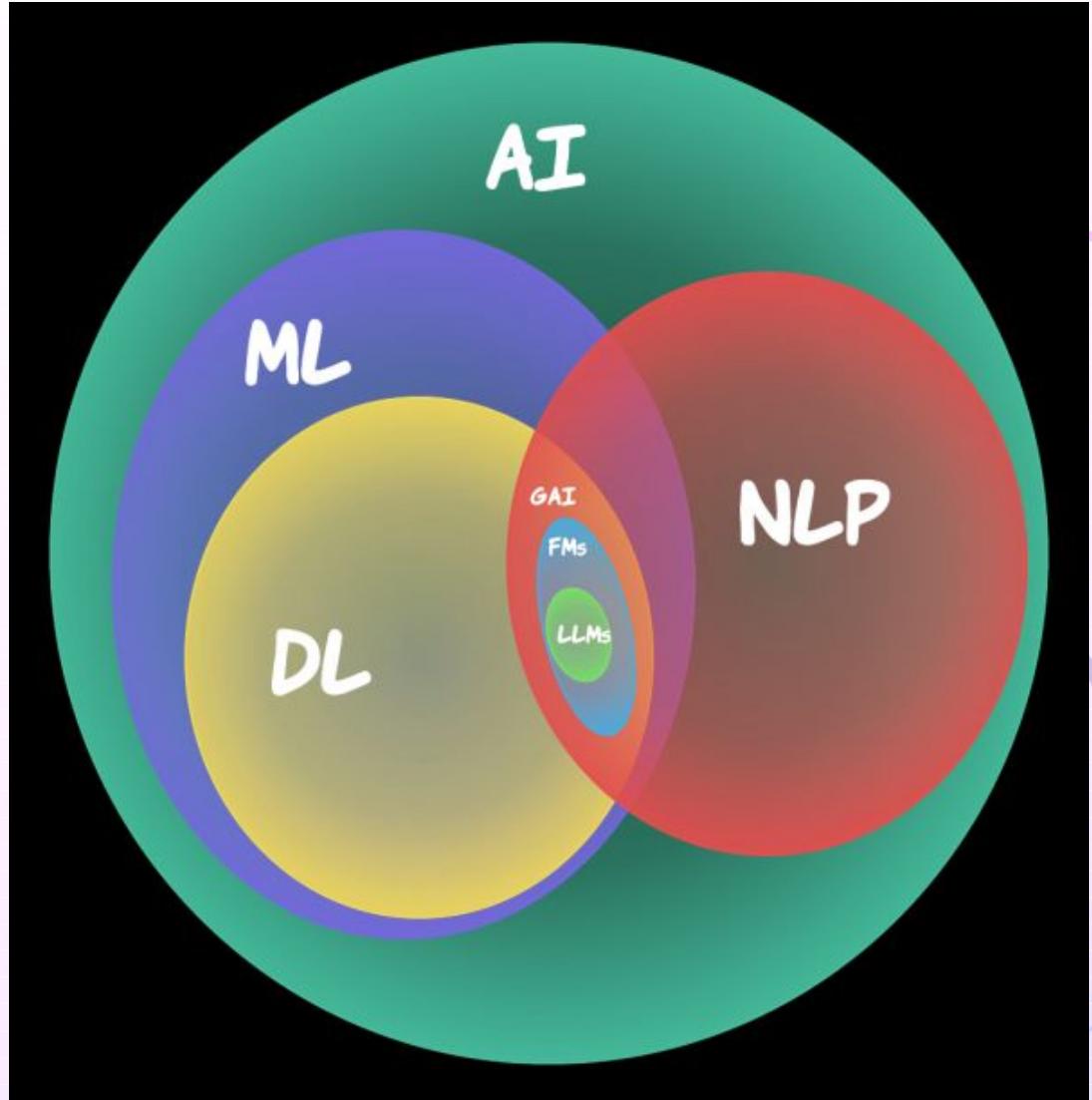
- GPT4
- Bard
- Llama
- BERT by Hugging Face

Large Language Models

GAI = Generative AI

FM = Foundational Models

Foundation models are more general, while LLMs are more focused on language.



Large Language Models - RAG

RAG = Retrieval Augmented Generation

RAG combines retrieval and generation processes to enhance the capabilities of LLMs

In RAG, the model retrieves relevant information from a knowledge base or external sources

This retrieved information is then used in conjunction with the model's internal knowledge to generate coherent and contextually relevant responses

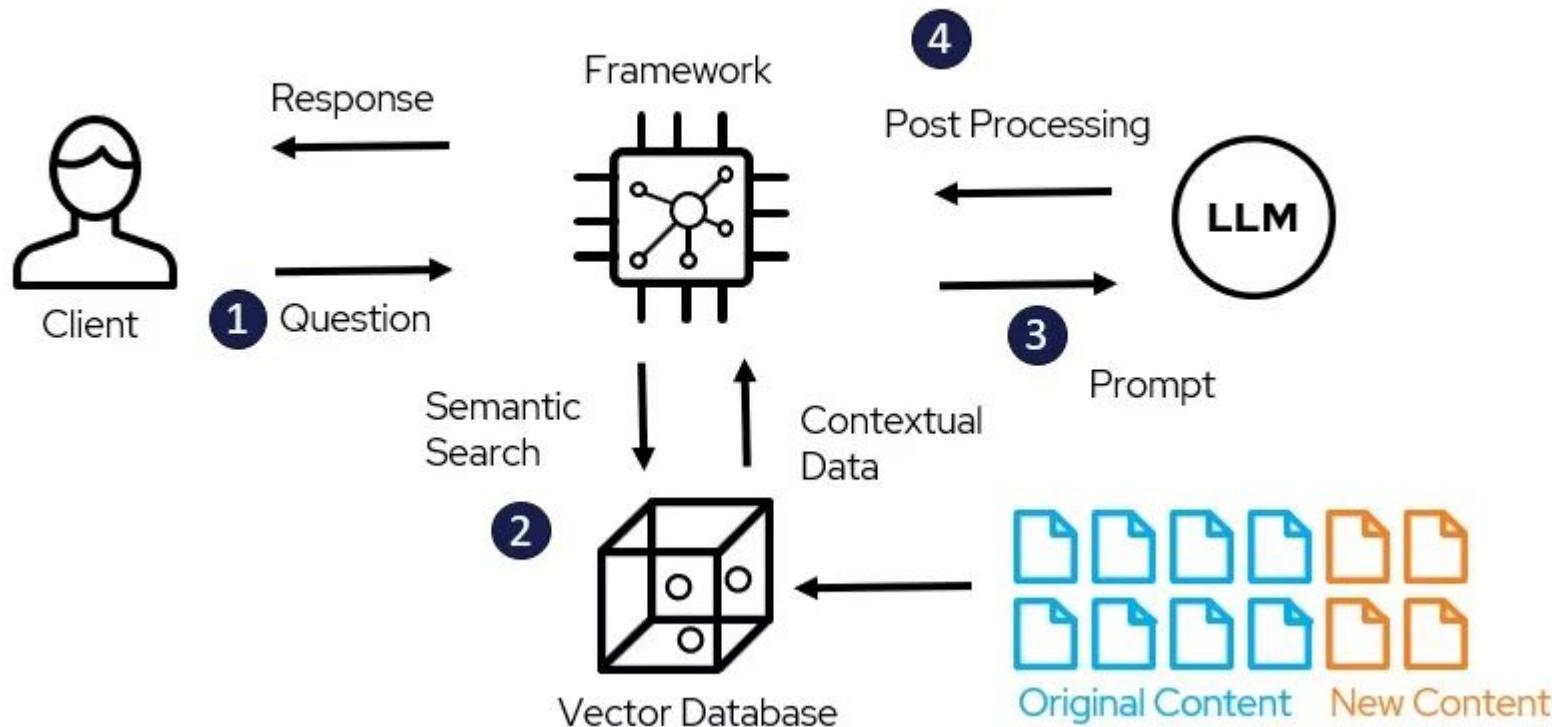
RAG enables LLMs to produce higher-quality and more context-aware outputs compared to traditional generation methods

Essentially, RAG empowers LLMs to leverage external knowledge for improved performance in various natural language processing tasks

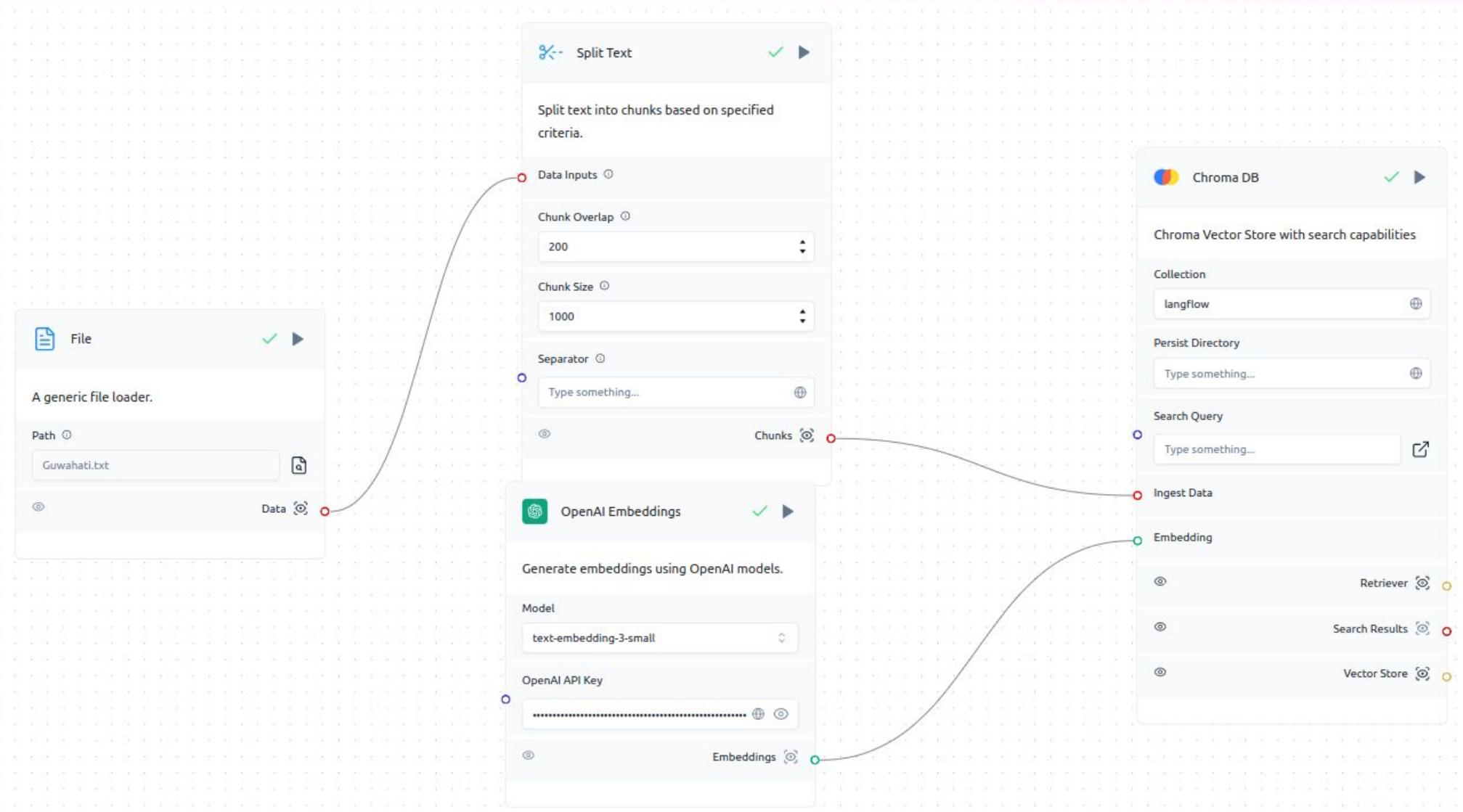
Large Language Models - RAG

1. The user submits a query to an LLM framework (like LangChain or LlamaIndex)
2. The RAG framework performs a semantic search of a vector database containing embeddings from the specified data sources (which can be updated with new content).
3. The results from vector database are submitted as a prompt to the LLM
4. The LLM framework formats the response from LLM and sends it back to the user. In the RAG model, the response is also likely to provide a link to some or all of the related source documents for the answer

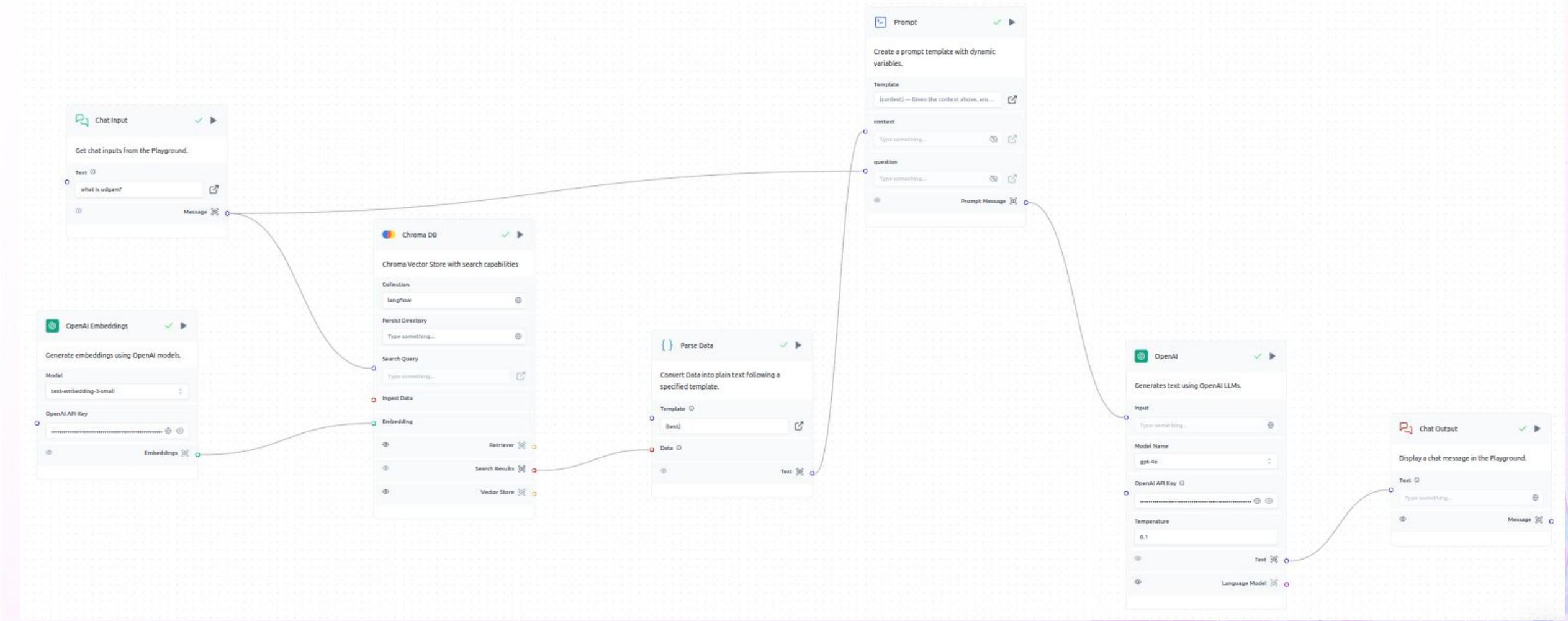
RAG Architecture Model



Large Language Models - RAG - Ingestion Workflow



Large Language Models - RAG Pipeline



Large Language Models - RAG Results

Playground 🚧

Interact with your AI. Monitor inputs, outputs and memories.

Memories

Default Session

The playground interface features a sidebar on the left with tabs for 'Memories' and 'Default Session'. The main area displays a conversation between a 'User' and an 'AI'. The user asks 'what is udgam?' and the AI provides two different responses. The first response is a general definition of the Sanskrit word 'udgam'. The second response is specific to the annual entrepreneurship summit at IIT Guwahati.

User
what is udgam?

AI
"Udgam" is a Sanskrit word that generally means "origin" or "source." It is often used in various contexts to denote the beginning or the point of emergence of something. In different cultural, spiritual, or academic discussions, "udgam" can refer to the origin of ideas, traditions, rivers, or even cosmic phenomena. The specific meaning can vary depending on the context in which it is used.

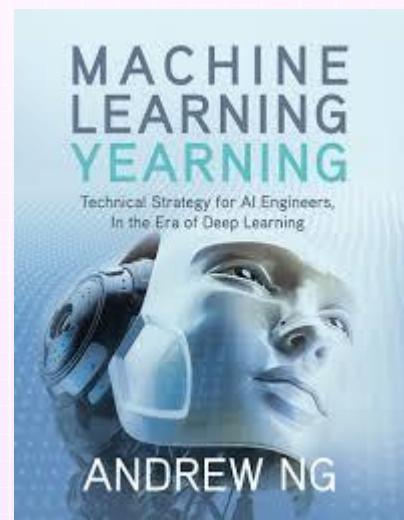
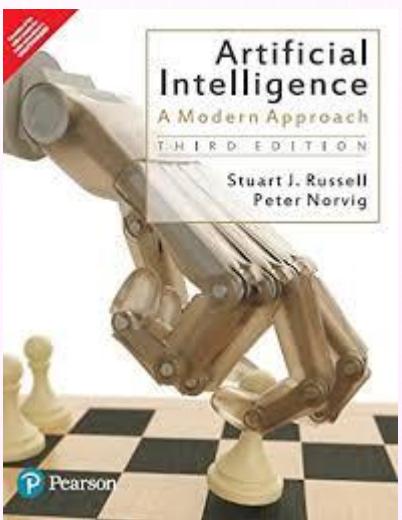
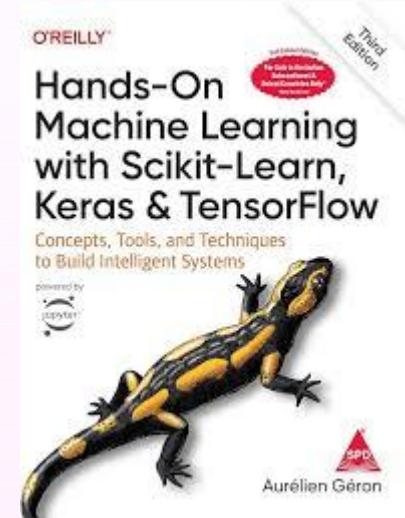
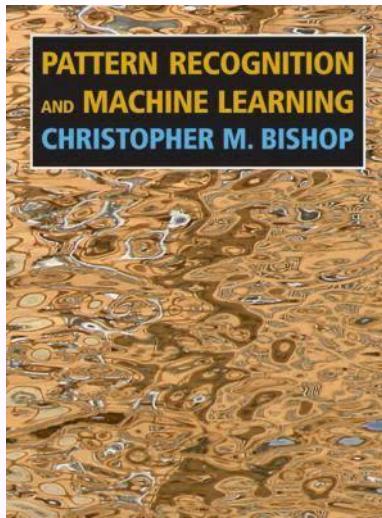
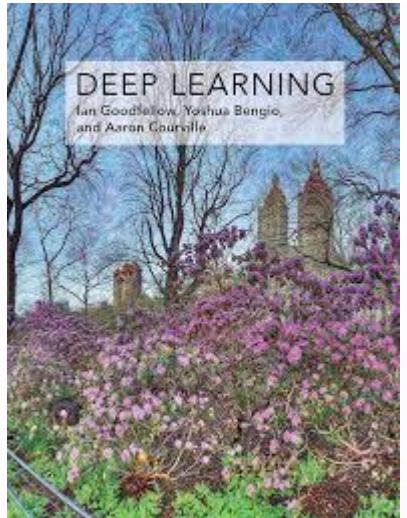
User
what is udgam?

AI
Udgam is the annual Entrepreneurship Summit of the Indian Institute of Technology (IIT) Guwahati. Organized by the institute's Entrepreneurial Cell, Udgam aims to inspire young students and graduates of North-East India to embrace entrepreneurship. The summit features a series of talks by eminent entrepreneurs, as well as a variety of workshops and networking sessions. It is typically held as a three-day event in January each year.

Send a message...

A large, semi-transparent decorative graphic with abstract, flowing pink and blue lines on a light background grid is positioned on the right side of the screen.

Book Recommendations for ML



Island of Knowledge

As our island of knowledge grows, so does the shore of our ignorance.

shore of ignorance - the increasing amount of things we don't know

island of knowledge - all that we DO know that we didn't know before.

while acting on the knowledge of what we do know, we can continue to grow by exploring the shores of what we don't.



Remember to have fun along the way

Contact

Jose Anitt Goli

gjanitt@gmail.com

8767981729

Conclusion

Great career opportunities at Mobius..!!

Do your life's best work with us..

Always on the lookout for right talent

Come join us..! or Train with us!

Thank you.