graVITas'24

UNLEASH YOUR GENIUS

THE AI & ML CLUB

TAM-VIT

PRESENTS

DATA ALCHEMY 2.0

Journey from basics to advanced Machine Learning

~ Sahil Murhekar (Vice Chairman, TAM-VIT)

CONTENTS OF SESSION

We’re going to delve in the following topics

INTRODUCTION	Basics of python, pandas, numpy and pipelines.
DATA PRE-PROCESSING	All about cleaning, manipulation and processing of data.
SUPERVISED LEARNING	Model training on Labelled data and simple maths behind it.
UNSUPERVISED LEARNING	Model training on Unlabelled data, clustering and recommendation systems.
REINFORCEMENT LEARNING	All about AI and Sensors, Actuators and feedbacks.
NATURAL LANGUAGE PROCESSING	Delving into human language processing and Chatbots.
DEEP LEARNING	Working with neural networks, image processing using CV and recognitions.
BONUS CONCEPTS	Some more concepts need to be known.

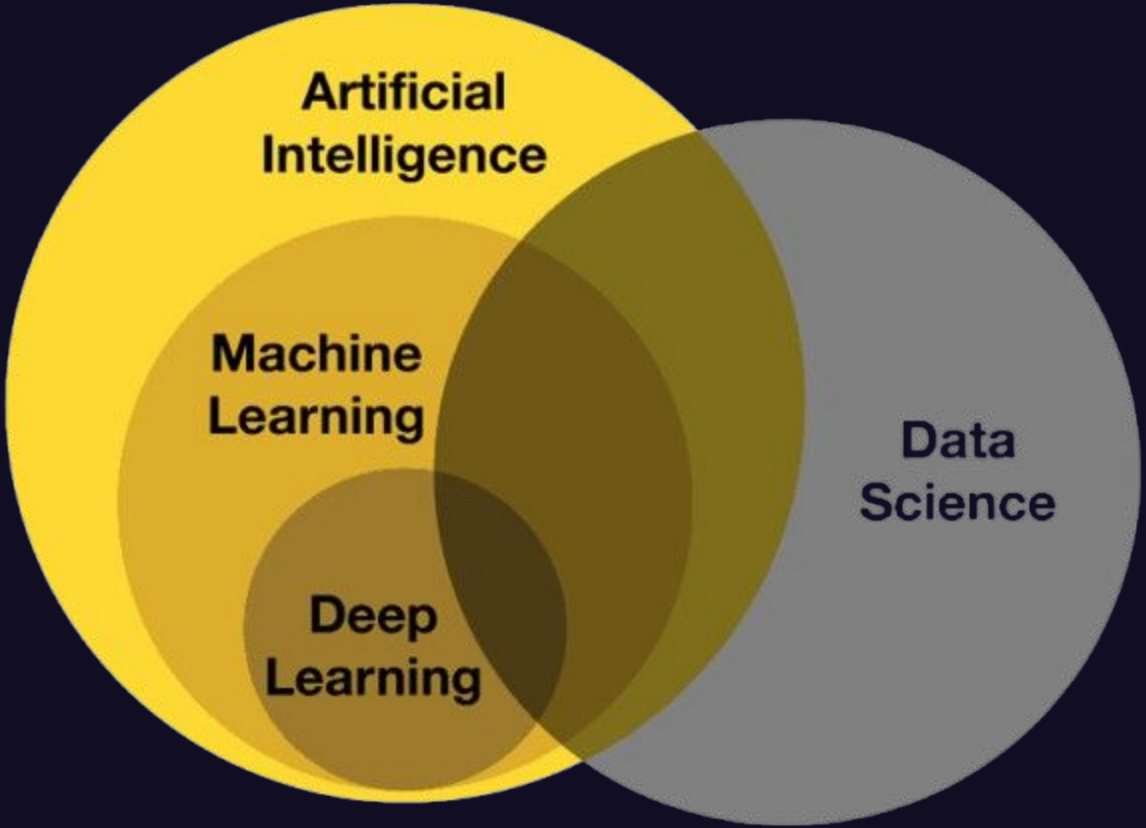
TABLE OF CONTENTS

AI will not take your jobs, but people using AI definitely will.

~ Jensen Huang (CEO, NVIDIA)

INTRODUCTIO N

01


Artificial Intelligence
Machine Learning
Deep Learning
Data Science

Statistics

- Mean , Median and Mode
- Variance and Std. deviation

$$\text{Variance}(\sigma^2) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \qquad \text{Standard Deviation}(\sigma) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

- Covariance and Correlation

$$\text{Correlation}(r) = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} \qquad (X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)$$

- Probability

$$P(A) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}} \qquad P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- Bayes Theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

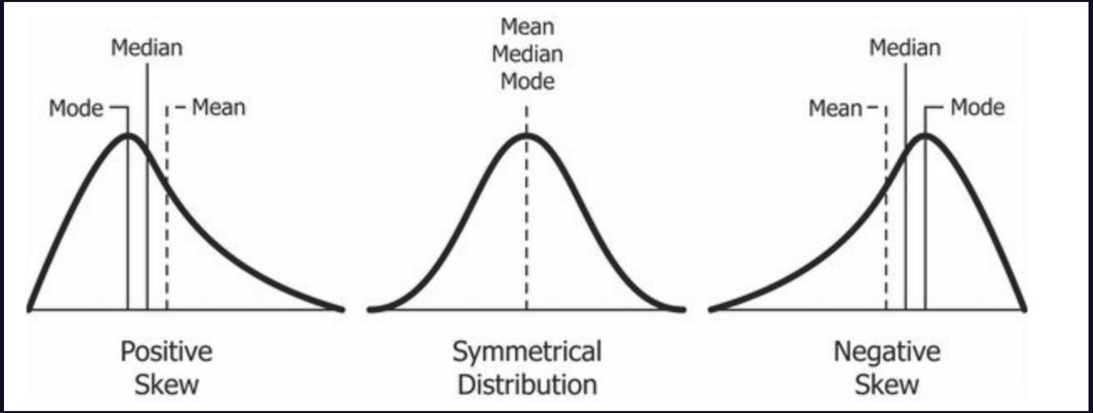
- Z - Score

$$Z = \frac{x - \mu}{\sigma}$$

Statistics

- Normal Distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Bias and Variance
  - Bias: Error due to overly simplistic assumptions.
  - Variance: Error due to model sensitivity to small fluctuations in the training set.

Basics of Python

Data Types

- Int
- float
- String
- boolean

Conditional Statements

- if/else

Loops

- For loop
- While loop

Storages

- Lists
- Tuples
- Sets
- Dictionary

File & Error Handling

- Opening file
- Closing file
- Writing and reading file
- Try and catch

Pandas

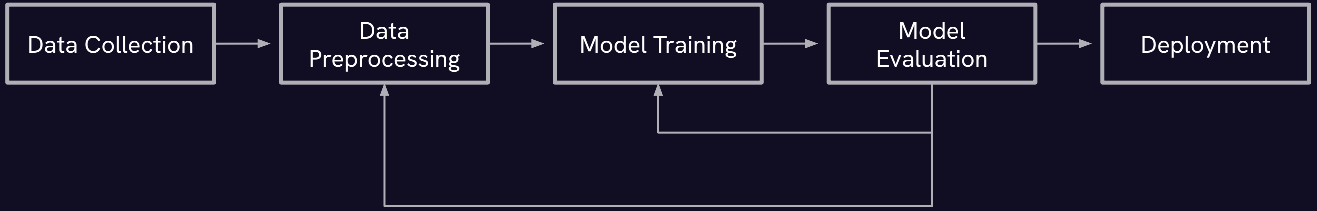
- Importing data
  - read_csv
  - read_excel
- Reading data
  - data.head()
  - data.tail()
  - data.info()
  - data.describe()
- Sorting
  - df.sort_values(by='column_name', ascending=False)
- Renaming columns
  - df.rename(columns={'old_name': 'new_name'}, inplace=True)
- Null values
  - data.isnull().sum()
  - data.fillna(x, inplace=True)
- Exporting data
  - df.to_csv('output.csv', index=False)
- Duplicates
  - data.drop_duplicates(inplace=True)
- Unique Values
  - data['column_name'].unique()
- Value counts
  - data['column_name'].value_counts()
- Date manipulation
  - data['year'] = data['date_column'].dt.year
  - data['month'] = data['date_column'].dt.month
  - data['day'] = data['date_column'].dt.day

Numpy

- Creating arrays
  - a = np.array([1, 2, 3])
  - b = np.array([[1, 2, 3], [4, 5, 6]])
  - zeros_array = np.zeros((3, 3))
  - ones_array = np.ones((2, 2))
  - identity_matrix = np.eye(3)
  - random_array = np.random.rand(2, 3)
- Operations
  - add = a + b
  - sub = a - b
  - mul = a * b
  - div = a / b
- Slicing
  - matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
  - row = matrix[1, :]
  - column = matrix[:, 1]
- Transpose
  - transpose = a.T

Pipelines

Machine Learning (ML) Pipelines are automated workflows used to streamline and manage the process of developing, deploying, and maintaining machine learning models.



DATA PRE-PROCESSIN G

02

Data Manipulation

NULL VALUES

ML Models work upon numeric data hence the null values of the dataset should be identified and cleared.

IMPUTATION

The process of inputting null values by others is called as imputation

NORMALISATION/ STANDARDISATION

Scaling the values in to a fixed range to minimise the processing time.

FEATURE EXTRACTION

Extracting new features from the existing data to make easy processing.

CORRELATION

Shows the relation and proportionality between data.

OUTLIERS

Checking the data points far away from the range to prevent overfitting or underfitting.

Types of Error Calculation

Mean Squared Error (MSE)

MSE is a popular metric used to measure the performance of regression models. It quantifies the average squared difference between the predicted and actual values. Lower MSE values indicate better performance.

Formula:
For a dataset with
n samples, where yi represents the actual values and y^i represents the predicted values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- y_i = actual value of the target variable for the i^{th} observation
- $\hat{y}_i$ = predicted value for the i^{th} observation
- n = total number of observations

R-squared Values

R-squared (also known as the coefficient of determination) measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges between 0 and 1, with a higher value indicating better model performance.

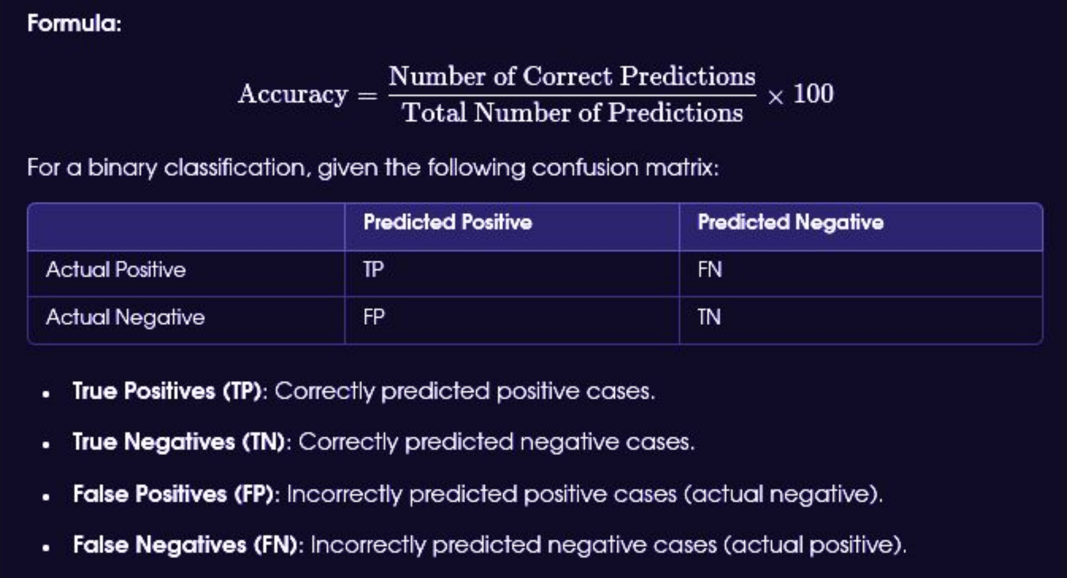
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- y_i = actual values
- $\hat{y}_i$ = predicted values
- $\bar{y}$ = mean of the actual values

Types of Error Calculation

Accuracy Score

Accuracy is used for classification tasks and is defined as the ratio of correctly predicted observations to the total observations. It is one of the simplest metrics for evaluating classification models.



Formula:

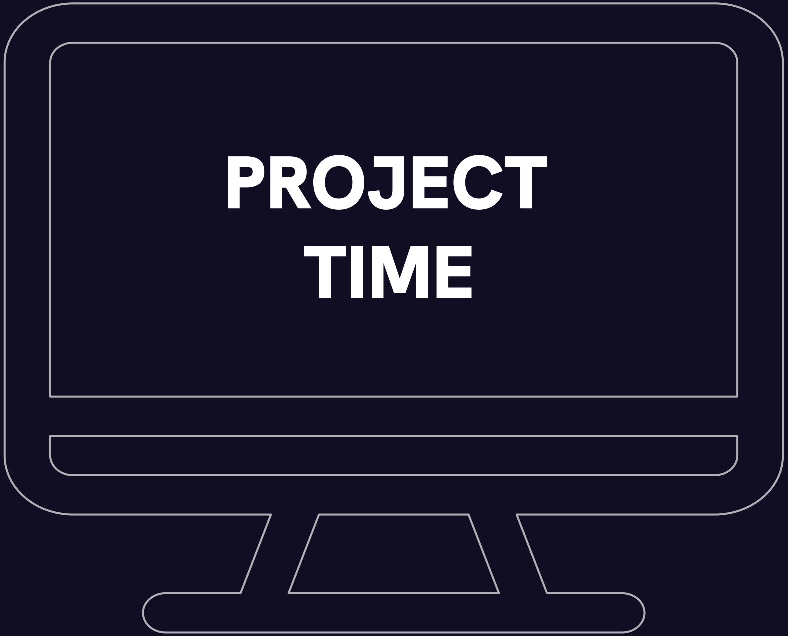
$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100$$

For a binary classification, given the following confusion matrix:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- **True Positives (TP):** Correctly predicted positive cases.
- **True Negatives (TN):** Correctly predicted negative cases.
- **False Positives (FP):** Incorrectly predicted positive cases (actual negative).
- **False Negatives (FN):** Incorrectly predicted negative cases (actual positive).

EDA (Exploratory Data Analysis)



PROJECT
TIME

SUPERVISED LEARNING

03

SUPERVISED LEARNING TYPES

REGRESSION

- Simple Linear
- Multiple linear
- Polynomial
- SVR (Support Vector Regression)
- Random Forest
- Decision Tree

CLASSIFICATION

- Logistic Regression
- KNN (K-Nearest Neighbors)
- SVM (Support Vector Machine)
- Kernel SVM
- Naive bayes
- Decision Tree
- Random Forest

REGRESSION

Simple Linear Regression

- Mathematics behind the model

  Simple linear regression fits a line to the data:

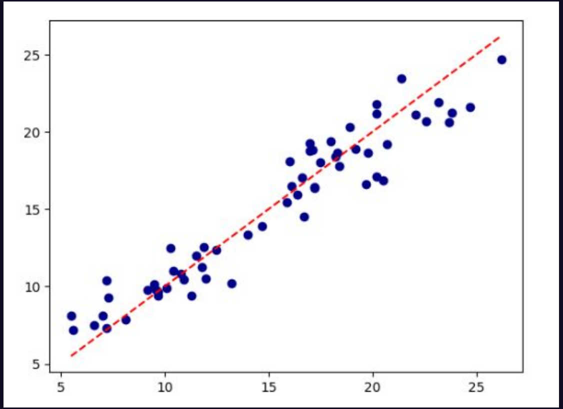
  $$Y = \beta_0 + \beta_1 X$$

- Code

```
from sklearn.linear_model import
LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

- Graph



Multipe Linear Regression

- Mathematics behind the model

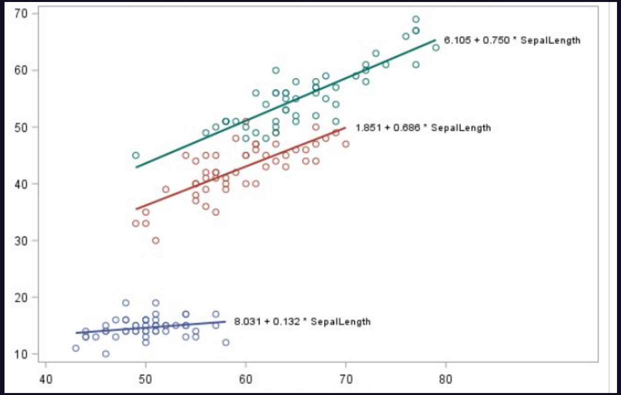
  Multiple linear regression uses several independent variables:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

- Code

```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

- Graph



Polynomial Linear Regression

- Mathematics behind the model

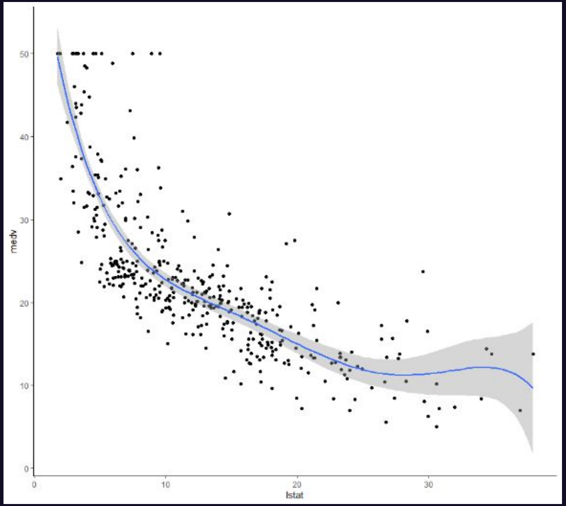
  Polynomial regression extends linear regression by including powers of the independent variable:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n$$

- Code

```
from sklearn.preprocessing import
PolynomialFeatures
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X_train)
lr.fit(X_poly, y_train)
```

- Graph



Support Vector Regression

- Mathematics behind the model

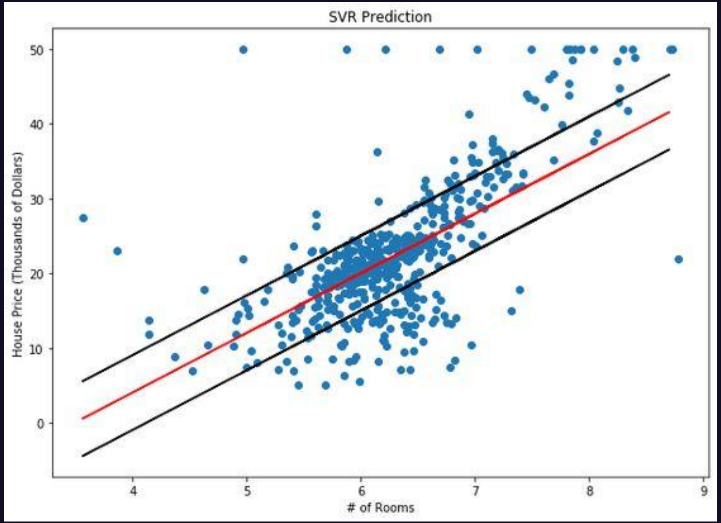
  SVR tries to fit a line within a margin (epsilon) around the data points:

$$\min \frac{1}{2}\|w\|^2 \text{ subject to } |y_i - (w \cdot x_i + b)| \leq \epsilon$$

- Code

```
from sklearn.svm import SVR
svr = SVR(kernel='rbf')
svr.fit(X_train, y_train)
y_pred = svr.predict(X_test)
```

- Graph



Decision Tree Regression

- Mathematics behind the model

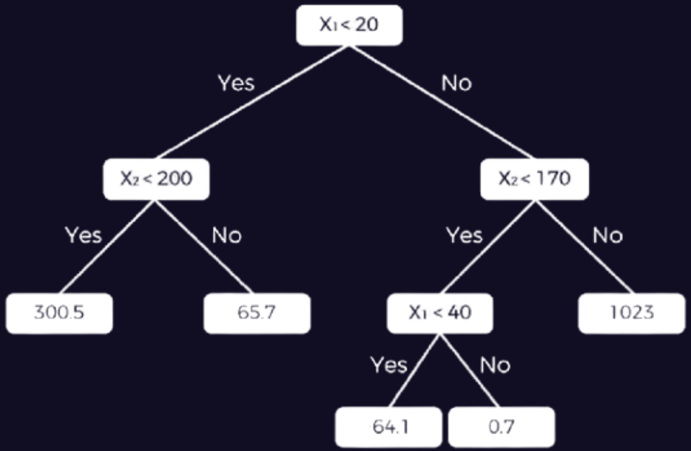
  Decision tree regression splits data based on minimizing the mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Code

```
from sklearn.tree import
DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
```

- Graph



Random Forest Regression

- Mathematics behind the model

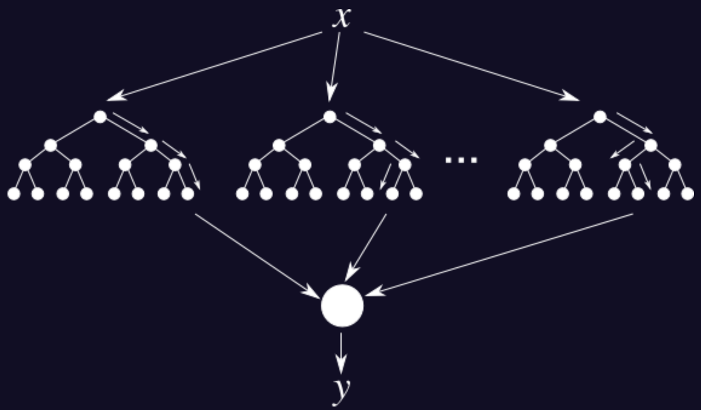
  Random forest regression averages the predictions of multiple decision trees:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(X)$$

- Code

```
from sklearn.ensemble import
RandomForestRegressor
rf =
RandomForestRegressor(n_estimators=100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

- Graph



CLASSIFICATIO N

Logistic Regression

- Mathematics behind the model

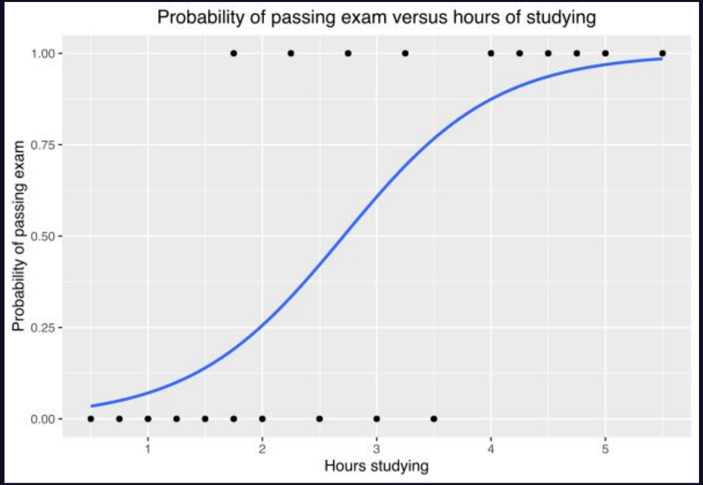
  Logistic regression predicts the probability of an outcome belonging to a particular class. The equation is:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}}$$

- Code

```
from sklearn.linear_model import
LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- Graph



K - Nearest Neighbors

- Mathematics behind the model

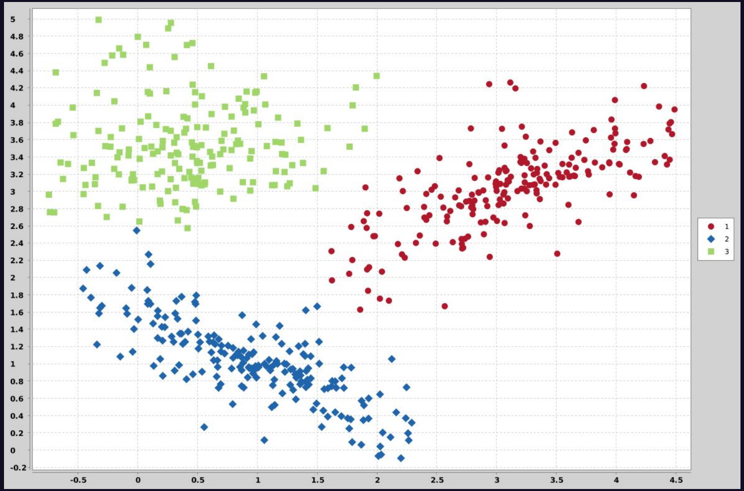
  KNN assigns a class based on the majority vote of the k-nearest neighbors in the feature space. The distance is usually calculated using Euclidean distance.

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

- Code

```
from sklearn.neighbors import
KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

- Graph



Support Vector Machine

- Mathematics behind the model

  SVM aims to find a hyperplane that maximizes the margin between two classes:

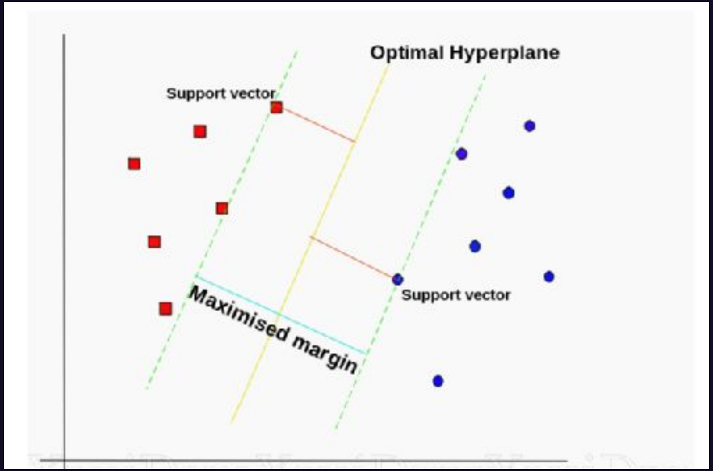
$$w \cdot x + b = 0$$

  where w is the weight vector and b is the bias term.

- Code

```
from sklearn.svm import SVC
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
```

- Graph



Kernel Support Vector Machine

- Mathematics behind the model

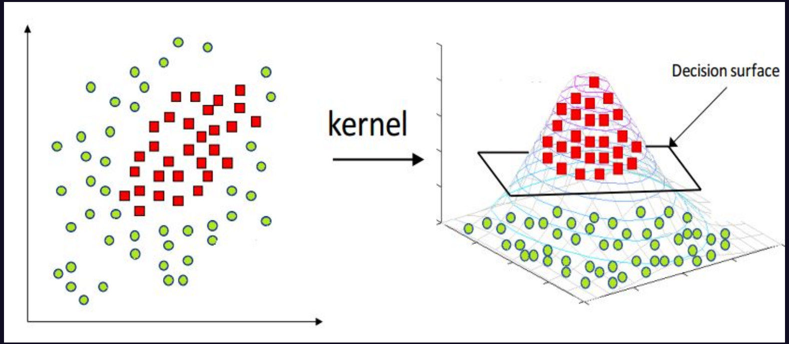
  Kernel SVM uses a kernel trick to map the input features into a higher-dimensional space, and a common kernel function is the radial basis function (RBF).

$$K(x, y) = e^{-\gamma\|x-y\|^2}$$

- Code

```
from sklearn.svm import SVC
kernel_svm = SVC(kernel='rbf')
kernel_svm.fit(X_train, y_train)
y_pred = kernel_svm.predict(X_test)
```

- Graph



Naive Bayes

- Mathematics behind the model

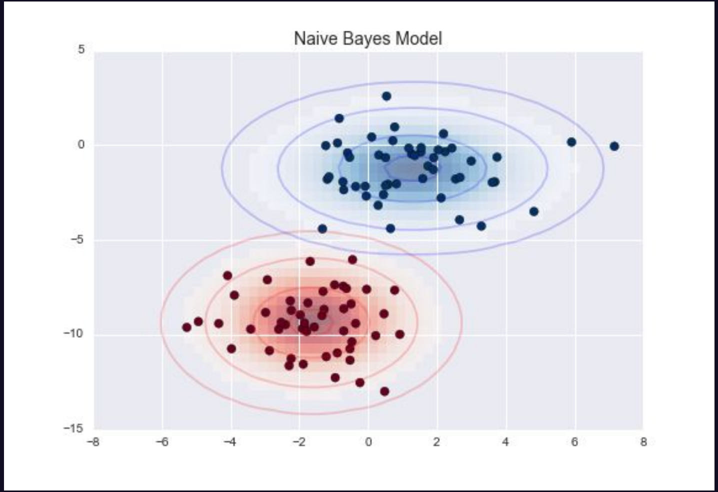
  Naive Bayes uses Bayes' theorem with the assumption of independence among features:

$$P(Y|X_1, X_2, \ldots, X_n) = \frac{P(Y)\prod_{i=1}^{n} P(X_i|Y)}{P(X_1, X_2, \ldots, X_n)}$$

- Code

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
```

- Graph



Decision Tree Classification

- Mathematics behind the model

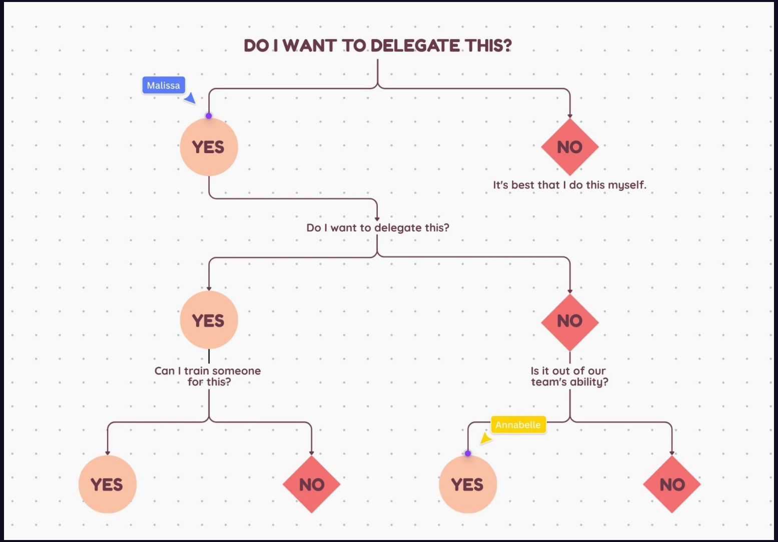
  Decision trees split the data based on the feature that provides the maximum information gain or minimizes the Gini impurity

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2$$

- Code

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
```

- Graph


DO I WANT TO DELEGATE THIS?
Malissa
YES
NO
It's best that I do this myself.
Do I want to delegate this?
YES
NO
Can I train someone for this?
Is it out of our team's ability?
Annabelle
YES
NO
YES
NO

Random Forest Classification

- Mathematics behind the model

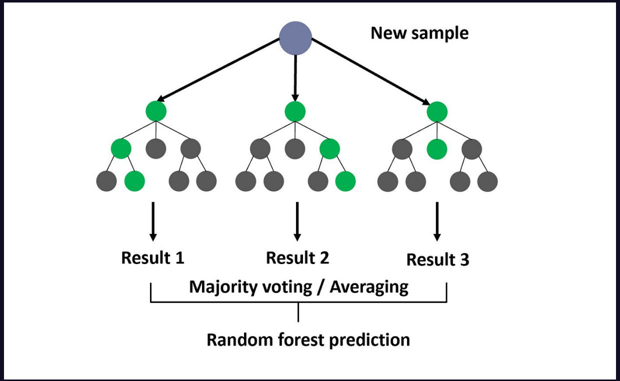
  Random forest builds multiple decision trees and combines their predictions (majority voting for classification)

$$\hat{y} = \mathrm{mode}(T_1(X), T_2(X), \ldots, T_B(X))$$

- Code

```
from sklearn.ensemble import
RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

- Graph



Cancer Data Predictions

PROJECT
TIME

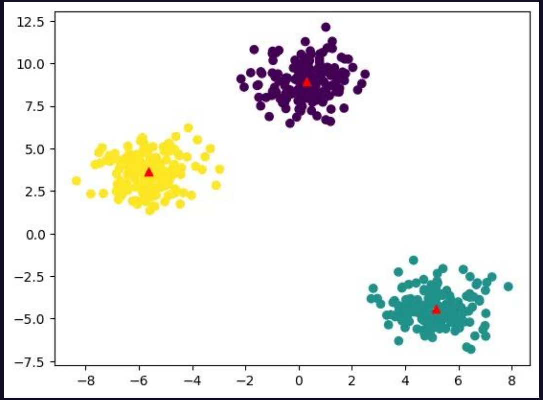
UNSUPERVISED LEARNING

04

K-Means Clustering

K-means clustering is an unsupervised machine learning algorithm that groups similar data points into a predetermined number (K) of clusters. It works by iteratively assigning data points to the nearest cluster center, then recalculating the center as the mean of its assigned points. This process repeats until the clusters stabilize. K-means is widely used for tasks like customer segmentation and image compression due to its simplicity and efficiency. However, it requires specifying the number of clusters in advance and performs best with spherical cluster shapes. The algorithm's effectiveness can be sensitive to the initial placement of cluster centers, sometimes necessitating multiple runs to find optimal groupings.

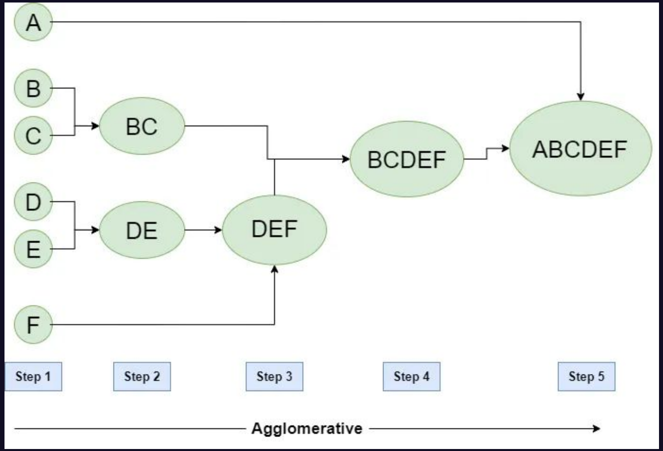
- Graph



Hierarchical Clustering

Hierarchical clustering is an unsupervised machine learning technique that creates a tree-like structure of data points based on their similarities. Unlike k-means, it doesn't require pre-specifying the number of clusters. The algorithm works in two main approaches: agglomerative (bottom-up) or divisive (top-down). In agglomerative clustering, each data point starts as its own cluster, and similar clusters are progressively merged. Conversely, divisive clustering begins with all points in one cluster and recursively splits them. The result is a dendrogram, a tree diagram representing the arrangement of clusters. This method allows for analysis at multiple levels of granularity and is particularly useful in fields like biology for creating taxonomies or in business for market segmentation. However, it can be computationally intensive for large datasets.

- Graph



Dimensionality Reduction

Dimensionality reduction is a technique in machine learning and statistics used to decrease the number of features in a dataset while preserving its essential characteristics. It addresses the "curse of dimensionality" where high-dimensional data can lead to overfitting and increased computational complexity. Common methods include Principal Component Analysis (PCA), which identifies the directions of maximum variance, and t-SNE, which is particularly effective for visualizing high-dimensional data in lower dimensions. Other approaches include feature selection, autoencoders, and Linear Discriminant Analysis (LDA). Dimensionality reduction not only simplifies data for analysis and visualization but can also improve model performance by reducing noise and redundancy. It's widely applied in fields such as image processing, bioinformatics, and natural language processing to extract meaningful patterns from complex, high-dimensional datasets.

PCA

Unsupervised method that reduces dimensionality by projecting data onto axes of maximum variance. Preserves global structure and is computationally efficient.

LDA

Supervised technique that reduces dimensions while maximizing class separation. Useful for classification tasks and works best with normally distributed classes.

Autoencoders

Neural networks that learn compressed data representations by reconstructing input through a bottleneck layer. Can capture non-linear relationships and are versatile for various data types.

Recommendation Systems

Algorithms that suggest items to users based on their preferences or behavior. Common types include collaborative filtering (user-item interactions), content-based filtering (item features), and hybrid approaches. Used widely in e-commerce, streaming services, and social media.

APRIORI

A classic algorithm for frequent itemset mining and association rule learning in databases. It uses a "bottom-up" approach, extending frequent subsets one item at a time. Efficient for large datasets but can be slow with many frequent itemsets.

ECLAT

(Equivalence Class Transformation) Another algorithm for frequent itemset mining. Uses a depth-first search strategy and a vertical database layout. Generally faster than Apriori for many datasets, especially those with long frequent patterns.

Movie Recommendations

PROJECT
TIME

REINFORCEMENT LEARNING

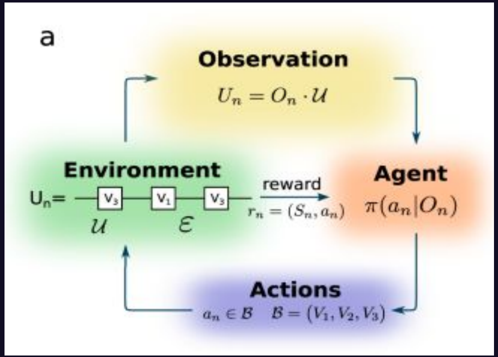
05

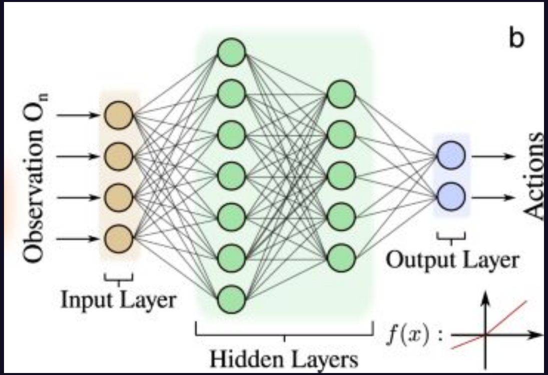
Reinforcement Learning

A machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent performs actions, receives rewards or penalties, and aims to maximize cumulative rewards over time. Key components include:

- Agent: The decision-maker
- Environment: The world the agent interacts with
- State: Current situation of the agent
- Action: Choices available to the agent
- Reward: Feedback from the environment
- Policy: Strategy for selecting actions

RL is used in game playing, robotics, autonomous vehicles, and resource management. Popular algorithms include Q-learning and Policy Gradient methods. Unlike supervised learning, RL doesn't require labeled data, but can be challenging due to the exploration-exploitation trade-off and delayed rewards.


a
Observation
$U_n = O_n \cdot \mathcal{U}$
Environment
$U_n=$ V₃ V₁ V₃
$\mathcal{U}$ $\mathcal{E}$
reward
$r_n = (S_n, a_n)$
Agent
$\pi(a_n|O_n)$
Actions
$a_n \in \mathcal{B}$ $\mathcal{B} = (V_1, V_2, V_3)$



NATURAL LANGUAGE PROCESSING

06

NLP Pipeline

STOPWORDS

Common words like "the," "and," or "is" are often removed from text to focus on the more informative content during text processing, as these words typically carry little meaning for tasks like text classification or sentiment analysis.

SEGMENTATION

The process of dividing text into smaller, meaningful units, such as sentences or paragraphs, to help models better understand the structure of the content.

TOKENIZATION

Splitting text into smaller units, or tokens, which can be words, subwords, or even characters. Tokenization is a crucial preprocessing step in NLP as it helps convert text into a format that models can process.

STEMMING

A preprocessing technique that reduces words to their base or root form (e.g., "playing" to "play"). This can help group similar words together during text analysis.

NLP Pipeline (cntd.)

LEMMATIZATION

Similar to stemming but more sophisticated. Lemmatization reduces words to their dictionary form (e.g., "better" to "good") by considering the context and ensuring the output is a valid word.

TRAINING

The process of teaching an NLP model by providing it with large datasets and letting it learn patterns, grammar, and semantics. Deep learning techniques such as neural networks are often used to build these models.

Fine-tuning

Fine-tuning is the process of taking a pre-trained language model and further training it on a smaller, task-specific dataset. This allows the model to adapt to specialized tasks (e.g., sentiment analysis, translation) without starting from scratch.

- **Process:**
  - A pre-trained model, such as BERT or GPT, is updated by training it on a dataset specific to the target task. Fine-tuning generally requires much less data and time than training a model from scratch since the model has already learned general language representations.

- **Advantages:**
  - Highly efficient as it builds on the vast knowledge of the pre-trained model.
  - Allows models to be customized for specific applications or domains (e.g., medical, legal, or technical language).

RAGs (Retrieval Augmented Generation)

RAG is a hybrid approach that integrates information retrieval (finding relevant external data) with language generation (text production by a model). In RAG, the model retrieves relevant documents from a large corpus of text (such as a knowledge base or database) before generating its response. This is particularly useful for tasks where the model needs to produce factually accurate, up-to-date, or specific information not available in the pre-trained model.

- **Process:**
  - The model first performs a retrieval step where it fetches relevant pieces of text or knowledge from an external source (such as Wikipedia or databases). It then uses this retrieved information to generate a coherent and relevant output.

- **Advantages:**
  - Combines the strengths of information retrieval (accessing large, up-to-date external data) with the fluency of generative models (producing natural-sounding text).
  - Useful for tasks that require factual accuracy or domain-specific knowledge, such as question answering or customer support.

Fake News Detection

PROJECT

TIME

▶ DEEP LEARNING 07

Deep Learning Basics

Deep learning is a subset of machine learning that uses artificial neural networks (ANNs) with many layers (hence "deep") to model complex patterns in data. It excels in tasks like image recognition, natural language processing, and speech recognition. Deep learning models automatically learn to extract useful features from raw data, making them highly powerful for unstructured data like images or text.

- Key Idea: Deep learning involves using layers of neurons (or nodes) to progressively learn abstract representations of the input data. Each layer captures more complex features as the data passes through.

- Deep learning models represent data as vectors and matrices. For example, input data is often a matrix X, and each layer of the network performs matrix multiplications.
  For each layer in the neural network:

  $$Z = W \cdot X + b$$

  Where:
  Z is the input to the activation function (also called pre-activation).
  W is the matrix of weights.
  X is the input matrix (output of the previous layer).
  b is the bias vector.

Image Processing with CV

Image processing is a critical area in computer vision, where images are enhanced or manipulated to extract useful information. OpenCV is a popular library for performing these tasks with efficient and simple implementations.

Tuning Images

Tuning images involves adjusting properties like brightness, contrast, saturation, and sharpness to improve the image quality or adapt it for further processing.

- **Brightness and Contrast Adjustment:**
    - Brightness refers to how light or dark an image appears, while contrast adjusts the difference between the brightest and darkest parts of an image.
    - In OpenCV, you can adjust brightness and contrast using the convertScaleAbs function:
- **Gamma Correction:**
    - This technique adjusts the brightness of an image in a nonlinear manner, making it useful for improving underexposed or overexposed images.

Image Processing with CV

Color Correction

Color correction is essential for adjusting the colors in an image, often to make them appear more natural or suitable for specific tasks.

- **Histogram Equalization:**
  - This technique improves the contrast of an image by redistributing its pixel intensity values, resulting in a clearer image, especially useful for grayscale images.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization):**
  - Similar to histogram equalization but operates in small regions of the image. It enhances contrast without over-amplifying noise.

Image Processing with CV

Edge Detection

Edge detection is used to identify the boundaries of objects within an image, which is crucial for applications like object recognition and segmentation.

- **Sobel Operator:**
  - This is a basic edge detection technique that calculates the gradient of the image intensity. The Sobel operator highlights areas where the intensity changes rapidly, identifying edges.
- **Canny Edge Detection:**
  - Canny is one of the most popular edge detection techniques due to its robustness and accuracy. It works by detecting gradients and applying thresholding to highlight strong edges.

Image Processing with CV

OpenCV offers various tools for enhancing, analyzing, and manipulating images for practical applications.

- **Blurring:**
  - Blurring reduces noise and detail in an image. Gaussian blur is one of the most common methods.
- **Thresholding:**
  - Thresholding converts an image into a binary format, where pixels are either set to the maximum value or to zero, based on a threshold value. This is often used for segmentation.
- **Morphological Operations:**
  - These operations refine image structures. Two common operations are:
    - Erosion: Shrinks object boundaries.
    - Dilation: Expands object boundaries.
- **Contours:**
  - Contours are continuous curves joining all the points with the same intensity. They are used for shape analysis, object detection, and recognition.

Neural Networks

ANN (Artificial Neural Networks)

ANNs are basic neural networks made up of an input layer, one or more hidden layers, and an output layer. Each neuron in one layer is connected to every neuron in the next layer (fully connected).

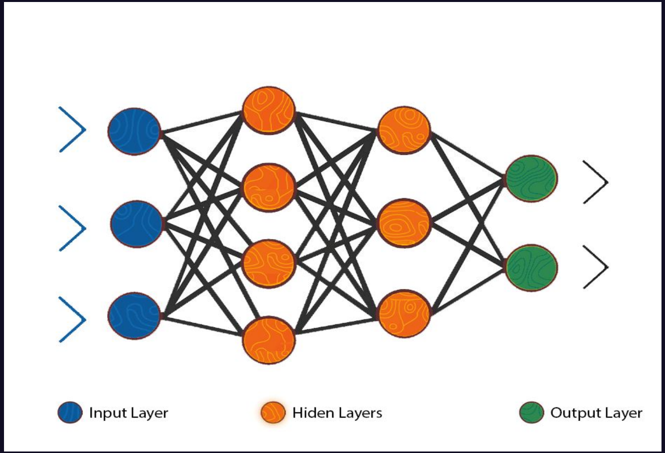
- **Use Case:** ANNs are used in various tasks like classification, regression, and prediction problems involving structured data.

Each neuron receives a weighted sum of inputs, applies an activation function, and passes the output to the next layer.
For each neuron:

$a=\sigma(W \cdot X+b)$

Where:
a is the output of the neuron (or activation).
σ is the activation function (e.g., ReLU, Sigmoid).

The goal of training is to minimize the loss function by adjusting W and b.



Neural Networks

CNN (Convolutional Neural Networks)

CNNs are specialized neural networks primarily used for image processing tasks. Instead of fully connected layers, CNNs use convolutional layers that apply filters to detect patterns like edges or textures in images.

- Key Components:
  - Convolutional layers extract features from the input image using filters.
  - Pooling layers reduce the spatial dimensions, retaining important information.
- Use Case: CNNs are widely used for image classification, object detection, and image segmentation.

CNNs introduce the concept of convolution:

$(I*K)(i,j)=\Sigma m \Sigma n I(i+m,j+n)\cdot K(m,n)$

Where:
I is the input image.
K is the filter or kernel (a matrix of weights).

The output is the result of sliding the filter over the input and performing element-wise multiplications.

After convolution, a pooling operation is often applied to reduce the dimensions, typically using max pooling.

MaxPooling(P)=max{P}MaxPooling(P)=max{P}

Where
P is the pooling region.

Examples for premade CNNs is U-Nets

Activation Functions

ReLU (Rectified Linear Unit)

Outputs the input if positive; otherwise, outputs 0. It's fast and works well in deep networks.

$$\mathrm{ReLU}(x) = \max(0, x)$$

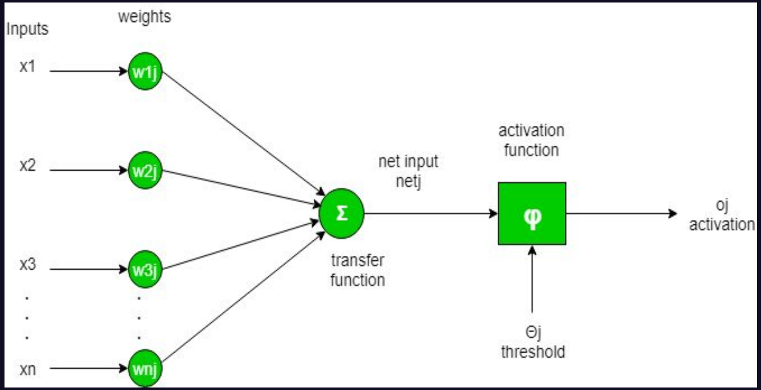
It outputs xxx if x>0x > 0x>0, otherwise 0. It's simple and computationally efficient, used in most deep networks.

Sigmoid Function

Squeezes the input into a range between 0 and 1, making it useful for binary classification tasks.

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

Squashes the output into the range (0, 1), useful for binary classification.



Activation Functions

Tanh (Hyperbolic Tangent)

Squashes the output between (-1, 1). It's symmetric around zero, unlike sigmoid.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Softmax:

Converts the output into a probability distribution over classes, often used in the output layer of multi-class classification problems.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Tensorflow Introduction

TensorFlow is an open-source machine learning library developed by Google. It is widely used for building and training deep learning models. TensorFlow provides a flexible and powerful framework to implement neural networks, including ANNs, CNNs, and more complex architectures like recurrent neural networks (RNNs) and transformers.

- Key Features:
  - Tensor operations: Data in TensorFlow is represented as multi-dimensional arrays (tensors).
  - GPU support: TensorFlow can use GPUs, which significantly speed up training deep learning models.
  - Keras: TensorFlow’s high-level API, Keras, simplifies model building by providing an intuitive interface for defining layers and training models.

Training and Epochs

TRAINING

Training refers to the process of adjusting the weights in a neural network by minimizing the error (loss) between predicted and actual outcomes.

- Steps of Training:
    - Forward pass: The input data is passed through the network, layer by layer, and the model makes predictions.
    - Loss calculation: The difference between the model's predictions and the true labels is calculated using a loss function.
    - Backpropagation: The error is propagated backward through the network, and weights are updated using an optimization algorithm like gradient descent to reduce the error.
    - Iteration: This process is repeated for multiple iterations, adjusting the weights to improve accuracy.

EPOCHS

- One epoch means that the entire dataset has been passed through the neural network once. In practice, multiple epochs are needed, as the model iteratively improves with each pass over the data.
- Example: If you have 1000 training examples and the batch size is 100, each epoch consists of 10 batches, and during training, the model sees the entire dataset 10 times.

FACE RECOGNITION

PROJECT
TIME

BONUS CONCEPTS

08

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are widely used in tasks like image generation, data augmentation, and style transfer. They consist of two main components:

- Generator: Creates fake data that mimics real-world data.
- Discriminator: Attempts to distinguish between real and generated data.

The two networks are trained together in a competitive process, where the generator tries to create realistic data to fool the discriminator, and the discriminator tries to improve its ability to detect fake data. Over time, the generator improves at creating realistic outputs, and the discriminator gets better at identifying them.

- Training Process:
  - Adversarial Training: The generator and discriminator are trained simultaneously. The generator learns to produce data that looks real, while the discriminator tries to classify whether the data is real or generated.
  - Applications: GANs are used for generating high-quality images, deep fake videos, image-to-image translation, and more.

- Challenges:
  - Mode Collapse: The generator might produce limited types of outputs.
  - Training Instability: GANs can be difficult to train due to the adversarial nature of their loss functions.

Variational AutoEncoders (VAEs)

Variational Autoencoders (VAEs) are another class of generative models that learn to represent data in a compressed format (latent space) and generate new data from it.

- Encoder: Compresses the input data into a lower-dimensional representation (latent vector).
- Decoder: Reconstructs the input from the latent vector.

Unlike regular autoencoders, VAEs generate data by learning a probability distribution over the latent space. This allows them to generate new data that closely resembles the original dataset.

- Key Features:
    - Latent Space: The learned latent space is continuous, meaning you can smoothly transition between different data points by moving in this space.
    - Probabilistic Model: VAEs provide more diversity in data generation compared to traditional autoencoders, since they model the uncertainty in the latent space.
    - Applications: VAEs are used in data generation, anomaly detection, and feature extraction.

- Challenges:
    - Blurry Outputs: The generated data might not be as sharp or realistic as GANs.
    - Less Realism: While VAEs offer structured latent spaces, the quality of generated data might not match the sharpness of GAN outputs.
-

U-Nets

Variational Autoencoders (VAEs) are another class of generative models that learn to represent data in a compressed format (latent space) and generate new data from it.

- Encoder: Compresses the input data into a lower-dimensional representation (latent vector).
- Decoder: Reconstructs the input from the latent vector.

Unlike regular autoencoders, VAEs generate data by learning a probability distribution over the latent space. This allows them to generate new data that closely resembles the original dataset.

- Key Features:
  - Latent Space: The learned latent space is continuous, meaning you can smoothly transition between different data points by moving in this space.
  - Probabilistic Model: VAEs provide more diversity in data generation compared to traditional autoencoders, since they model the uncertainty in the latent space.
  - Applications: VAEs are used in data generation, anomaly detection, and feature extraction.
- Challenges:
  - Blurry Outputs: The generated data might not be as sharp or realistic as GANs.
  - Less Realism: While VAEs offer structured latent spaces, the quality of generated data might not match the sharpness of GAN outputs.
-

tamvit

@tam_vit.vellore

TAM-VIT

THANK YOU !

~The AI & ML Club (TAM)