Clustering Algorithms

Clustering Algorithms

Clustering algorithms are a type of **unsupervised learning technique** used to group data points into clusters such that objects within the same cluster are more similar to each other than to those in other clusters.

Unlike classification, clustering does not rely on **predefined labels**; instead, it discovers the inherent grouping in the data.

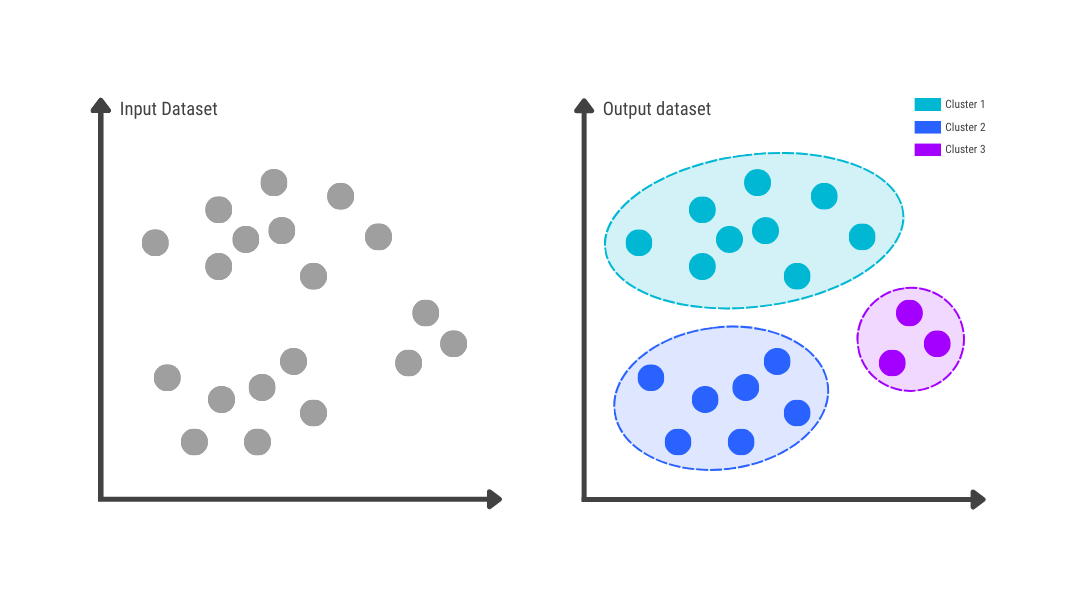
To determine cluster membership, most algorithms evaluate the distance between a point and cluster centroids.

The output of the clustering algorithm is a statistical description of the **cluster centroids** with the number of components in each cluster.

Cluster centroid is a point whose parameter values are the mean of the parameter values of all the points in the clusters

Desirable properties of clustering algorithms are:

1. Scalability (in terms of both time and space)
2. Ability to deal with different data types
3. Minimal requirements for domain knowledge to determine input parameters
4. Interpretability and usability



**Types of Clustering Algorithms**

1. Partitioning Clustering

Divides the data into **non-overlapping clusters**.

Example: K-Means Clustering

1. Hierarchical Clustering

Builds a **tree-like structure** (dendrogram) to represent data clusters

Example: Agglomerative Hierarchical Clustering

1. Density-based Clustering

Forms clusters based on the density of data points in the feature space.

Example: DBSCAN

1. Model-based Clustering

Assumes data is generated from a mixture of probability distributions.

Example: Gaussian Mixture Models (GMM)

K-Means Clustering

K-Means Clustering

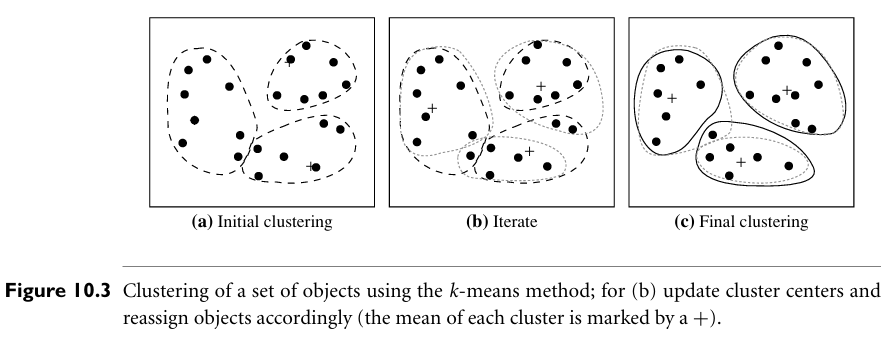
K-Means clustering is a popular unsupervised machine learning algorithm used for partitioning data into k distinct, non-overlapping groups (clusters).

The primary objective of K-Means is to minimize the variance within each cluster and maximize the variance between clusters. It works well when the data is numerical and the clusters are spherical.

In this method, each cluster is represented by the center of the cluster.

‘K’ - stands for the number of clusters.

It is a straightforward algorithm and efficient for large datasets as it typically converges quickly.



**Working of K-Means Clustering:**

1. Select the Number of Clusters (k)

* Specify the number of clusters (k) you want to form.
* The choice of k can be based on domain knowledge, the Elbow Method, or the Silhouette Score.

1. Initialize Centroids:

* Randomly select k points from the dataset as the initial centroids.
* These centroids represent the center of each cluster.

1. Assign Points to the Nearest Centroid:

* Calculate the Euclidean distance between each data point and each centroid.
* Assign each point to the nearest centroid, forming clusters.

1. Update Centroids:

* Calculate the mean of points in each cluster to get the new centroid.
* The new centroid is the average position of all the points in that cluster.

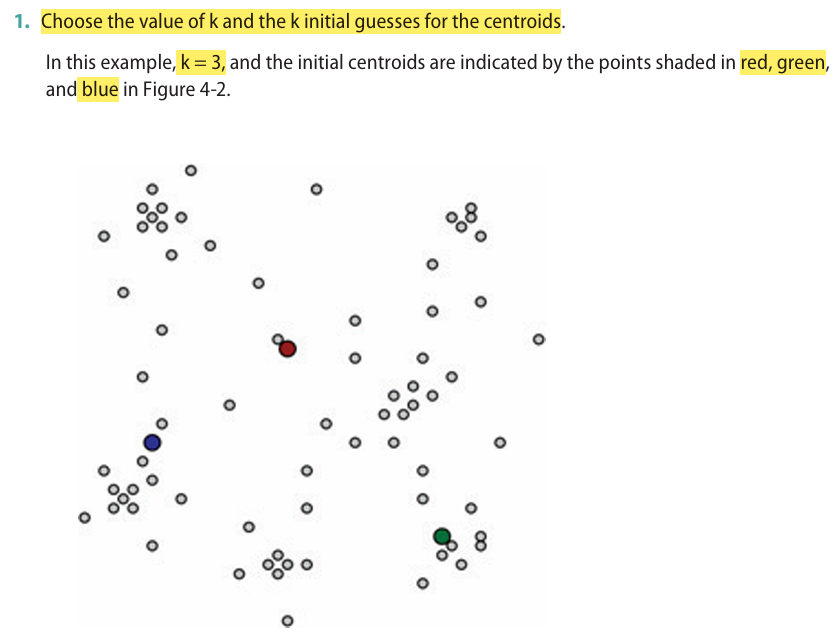
1. Repeat the Assignment and Update Steps:

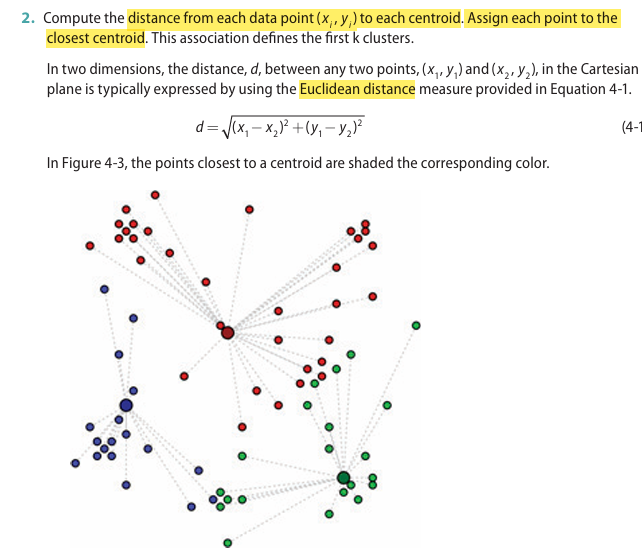
* Iterate the assignment and centroid update steps until Centroids do not change (convergence) or Maximum iterations are reached.

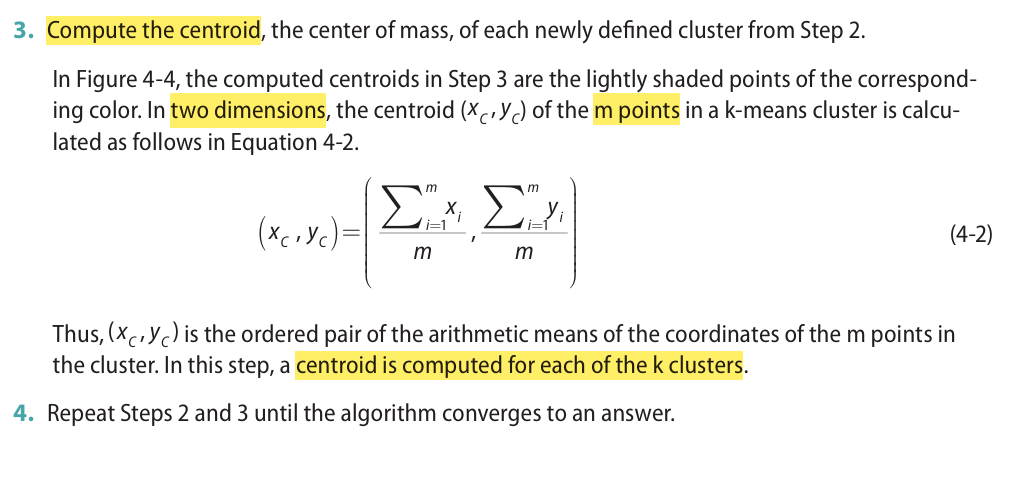
**Disadvantages of K-Means Clustering**:

1. Choice of k: Choosing the optimal number of clusters can be challenging.
2. Sensitive to Initial Centroids: Different initializations can result in different clusters.
3. Not Suitable for Non-Spherical Clusters: Cannot detect clusters with complex shapes.
4. Sensitive to Outliers: Outliers can significantly distort centroids.
5. Scales Poorly with High Dimensions: Distance calculations become less meaningful.

**Example of K-Means Clustering**







Hierarchical Clustering

Hierarchical Clustering

A hierarchical clustering method works by grouping data objects into a hierarchy or “**tree**” of clusters

Unlike K-means clustering, where the number of clusters (k) must be predefined, hierarchical clustering forms a tree-like structure called a **dendrogram**.

This method is particularly useful when the data has a nested or hierarchical structure.

Hierarchical clustering can be further classified as either **agglomerative** or **divisive** depending on hierarchical decomposition - bottom-up (merging) or top-down (splitting) fashion.

**Types of Hierarchical Clustering**

Agglomerative (Bottom-Up) Clustering:

* Starts with each data point as an individual cluster.
* Iteratively merges the closest pairs of clusters until only one cluster remains or the desired number of clusters is reached.
* Most commonly used method.

Divisive (Top-Down) Clustering:

* Starts with the entire dataset as one cluster.
* Iteratively splits the cluster into smaller sub-clusters.
* Less common compared to agglomerative clustering.

**Working of Hierarchical Clustering**:

The following steps explain the agglomerative hierarchical clustering process:

Step 1: Calculate Distance Matrix:

* Compute the pairwise distances between all data points using metrics like Euclidean distance or Manhattan distance.

Step 2: Create Individual Clusters:

* Treat each data point as a separate cluster.

Step 3: Merge Clusters:

* Merge the two closest clusters to form a new cluster.
* Update the distance matrix to reflect the new cluster distances.

Step 4: Update Distances (Linkage Methods):

* Choose a linkage criterion to update the distance between clusters:

**Single Linkage**: Minimum distance between points in two clusters.

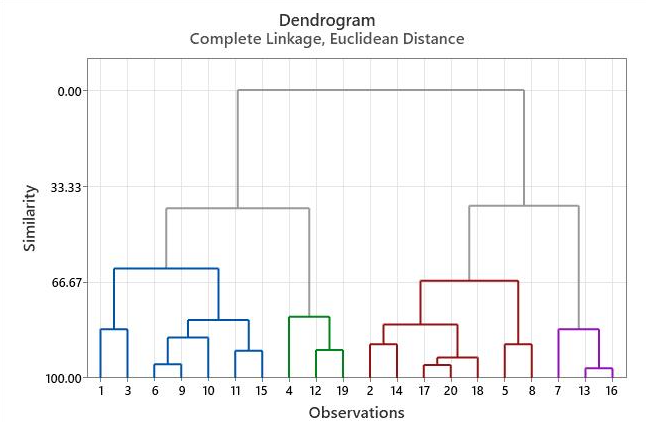
**Complete Linkage**: Maximum distance between points in two clusters.

**Average Linkage**: Mean distance between points in two clusters.

**Centroid Linkage**: Distance between cluster centroids.

Step 5: Repeat Until One Cluster Remains:

* Continue merging clusters based on the chosen criterion until only one cluster or the desired number of clusters is left.

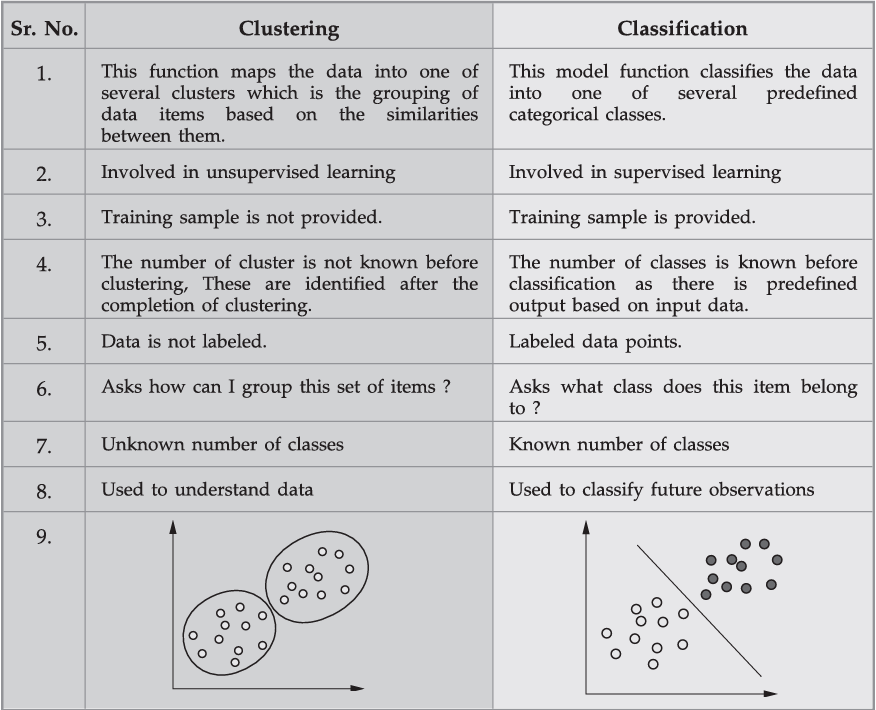


### **Real-World Applications of Hierarchical Clustering**

1. **Genomics and Bioinformatics**
   * Grouping genes or proteins based on expression levels.
   * Understanding evolutionary relationships (phylogenetic trees).
2. **Market Segmentation**
   * Grouping customers based on buying behavior, demographics, or preferences.
3. **Document Clustering**
   * Organizing news articles, research papers, or web pages into thematic groups.
4. **Social Network Analysis**
   * Identifying communities or subgroups in a network.
5. **Image Segmentation**
   * Breaking down an image into hierarchical regions based on color or texture similarity.

Clustering vs Classification

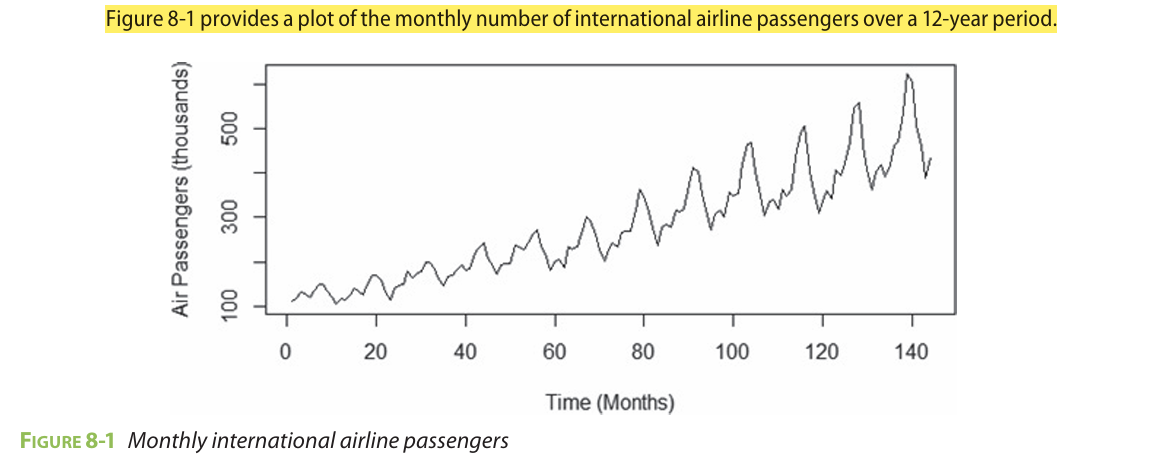
Clustering vs Classification



Time Series Analysis

Time Series Analysis

1. A **time series** is a sequence of data points collected or recorded at successive time intervals. In other words, it represents data points that are measured and ordered in time.
2. **Time series analysis** is a statistical technique used to analyze these time-ordered data points collected at consistent intervals.
3. The primary goal is to identify patterns, trends, and other characteristics within the data to make accurate predictions and insights.



1. A time series can consist of the following components - Trend, Seasonality, Cyclic and Irregularity
2. The **trend** refers to the long-term movement in a time series. It indicates whether the observation values are increasing or decreasing over time.
3. The **seasonality** component describes the fixed, periodic fluctuation in the observations over time.
4. The **cyclical** component also refers to a periodic fluctuation, but one that is not as fixed
5. The **irregularity** component describes random, unpredictable fluctuations.

Techniques used for time series analysis:

1. ARIMA model
2. Box-Jenkins multivariate models
3. Holt winters exponential smoothing

Applications of Time Series Analysis:

1. Financial Forecasting: Predicting stock prices and economic trends.
2. Weather Prediction: Analyzing temperature and climate patterns.
3. Healthcare Monitoring: Tracking patient vital signs over time.
4. Sales and Marketing: Forecasting sales for inventory planning.
5. Energy Consumption: Predicting electricity usage patterns.

ARIMA Model

ARIMA Model

1. ARIMA - AutoRegressive Integrated Moving Average
2. It is a popular statistical method used for time series forecasting.
3. It is particularly effective for analyzing and predicting data that exhibit patterns over time, such as stock prices, economic indicators, or weather patterns.
4. Its main application is in the area of short term forecasting requiring at least 40 historical data points and works best when data exhibits a stable or consistent pattern over time with minimum amount of outliers
5. Parameters of ARIMA model:

* **p** (lag order) : Number of lag observations included in model
* **d** (degree of differencing) : Number of times the raw observations are differenced.
* **q** (order of moving average) : Size of the moving average window.

1. Assumptions of ARIMA model:

* **Data should be stationary:**

Time series is stationary if

1. The expected value (mean) of is constant for all values of t
2. The variance of is finite
3. The covariance of and depends only on the value of h = 0,1,2,..for all t

* **Data should be univariate:** It works only on a single variable.

1. Components of ARIMA

* **AR** (AutoRegressive)

Uses the dependent relationship between an observation and a number of lagged observations.

Represented as AR(p) where p is the number of lagged observations (lags).

Example: If p=2, the model considers the past two time points

𝑌𝑡 = Current value

𝑐 = Constant

𝜙 = Coefficients

𝜖t = Error term

* **I** (Integrated) part:

Represents the differencing of raw observations to make the time series stationary (constant mean and variance).

Represented as I(d) where d is the number of differencing steps required.

Example: If d = 1, the difference between consecutive observations is used.

* **MA** (Moving Average)

Uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Represented as MA(q) where q is the size of the moving average window.

The ARIMA(p, d, q) model is a combination of:

p: Number of lag observations included (AR part).

d: Degree of differencing to make data stationary (I part).

q: Size of the moving average window (MA part).

An ARIMA(2, 1, 1) model has:

2 lagged values,

1 difference to ensure stationarity,

1 moving average component.

**Advantages of ARIMA:**

* Effective for short-term forecasting when data shows linear trends.
* Handles both autoregressive and moving average patterns.
* Efficient for univariate time series data.

**Limitations:**

* Ineffective for non-linear time series.
* Assumes that future values are linearly dependent on past values.
* Requires data to be stationary, which may involve complex transformations.

k-fold Cross Validation & Random Subsampling

Explain k-fold Cross Validation & Random Subsampling.

### **1. K-Fold Cross Validation**

**Definition**:  
 K-Fold Cross Validation is a **model evaluation technique** where the dataset is divided into **K equal parts (folds)**. The model is trained on **K−1 folds** and tested on the remaining **1 fold**. This process is repeated **K times**, each time using a different fold as the test set.

**How it works**:

1. Divide the dataset into **K equal folds**.
2. For each of the K iterations:  
   * Use **K−1 folds** for training.
   * Use the remaining **1 fold** for testing.
3. Compute the **evaluation metric (e.g., accuracy, RMSE)** each time.
4. **Average the results** over all K iterations for a final performance estimate.

**Example**:

* If K = 5 (i.e., 5-Fold CV), the data is split into 5 parts.
* The model is trained 5 times, each time leaving out one different fold for testing.

**Advantages**:

* Provides a **more reliable estimate** of model performance.
* Makes **better use of data**, especially in small datasets.
* Reduces variance due to random sampling.

**2. Random Subsampling (Monte Carlo Cross Validation)**

**Definition**:  
 Random Subsampling involves **randomly splitting the data** into **training and test sets** multiple times, and **averaging the evaluation results**. Unlike K-fold CV, the splits are not necessarily disjoint.

**How it works**:

1. For each iteration:  
   * Randomly split the dataset into a **training set** and a **test set** (e.g., 70%–30%).
   * Train the model on the training set and test on the test set.
2. Repeat the process **N times** (e.g., 10 or 20 iterations).
3. **Average the performance metrics** across all runs.

**Advantages**:

* Easy to implement and flexible with different split ratios.
* Less sensitive to the choice of a particular data split.

**Disadvantages**:

* The test sets may **overlap** across iterations.
* Does **not guarantee** every data point is tested.

**Comparison Table**

| **Aspect** | **K-Fold Cross Validation** | **Random Subsampling** |
| --- | --- | --- |
| Data Splitting | K disjoint folds | Random splits (may overlap) |
| Coverage | Every point is used for testing once | Some points may never be tested |
| Repetition | Fixed K iterations | Arbitrary number of repetitions (N) |
| Accuracy of Estimate | Usually more stable | Can vary depending on split |
| Computational Cost | Moderate (K model trainings) | Can be higher with more iterations |

Confusion Matrix

Confusion Matrix

### **📘 Confusion Matrix – Explained**

A **confusion matrix** is a performance evaluation tool for **classification models**, especially in **supervised learning**. It helps visualize how well the model is predicting by comparing **actual values** with **predicted values**.

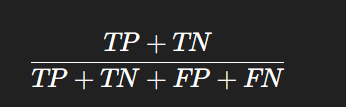
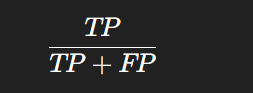
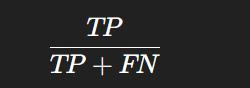
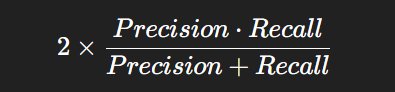
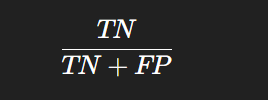
### **✅ Structure of a Confusion Matrix (for Binary Classification)**

|  | **Predicted: Positive** | **Predicted: Negative** |
| --- | --- | --- |
| **Actual: Positive** | True Positive (TP) | False Negative (FN) |
| **Actual: Negative** | False Positive (FP) | True Negative (TN) |

### **📌 Terminology**

* **True Positive (TP)**: The model correctly predicted **positive** (e.g., detected a disease when the patient has it).
* **True Negative (TN)**: The model correctly predicted **negative**.
* **False Positive (FP)**: Type I error – model predicted **positive** when the actual was negative.
* **False Negative (FN)**: Type II error – model predicted **negative** when the actual was positive.

### **📊 Performance Metrics Derived from the Confusion Matrix**

1. **Accuracy** =  
     
    How often the model is correct.
2. **Precision** =  
      
    Out of all positive predictions, how many were correct?
3. **Recall (Sensitivity or TPR)** =  
     
    Out of all actual positives, how many were predicted correctly?
4. **F1-Score** =  
     
    Harmonic mean of precision and recall.
5. **Specificity (TNR)** =  
      
    Out of all actual negatives, how many were correctly predicted?

### **✅ Why Use a Confusion Matrix?**

* It shows **not just overall accuracy**, but **where the model is making errors**.
* Helps identify **imbalanced class issues** (e.g., more false negatives than false positives).
* Useful in domains like **medical diagnosis**, **fraud detection**, or **spam filtering** where the **cost of wrong prediction matters**.

### **🧠 Example**

If a model is tested on 100 samples and gives:

* TP = 50, TN = 40, FP = 5, FN = 5

Then:

* Accuracy = (50+40)/100 = **90%**
* Precision = 50 / (50+5) = **90.91%**
* Recall = 50 / (50+5) = **90.91%**
* F1 Score ≈ **90.91%**

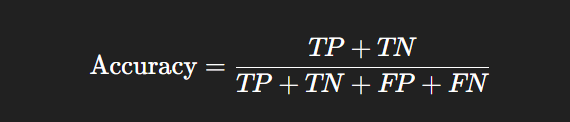
Explain the terms:

Explain the terms:

### **i) Accuracy**

**Definition**:  
 Accuracy measures the **overall correctness** of the model. It is the ratio of correctly predicted observations (both true positives and true negatives) to the total observations.

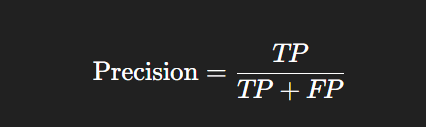
**Formula**:



### **ii) Precision**

**Definition**:  
 Precision measures how many of the predicted positive results are **actually correct**. It focuses on the quality of **positive predictions**.

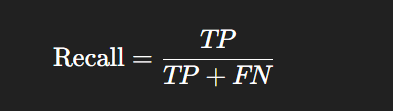
**Formula**:



### **iii) Recall (Sensitivity or True Positive Rate)**

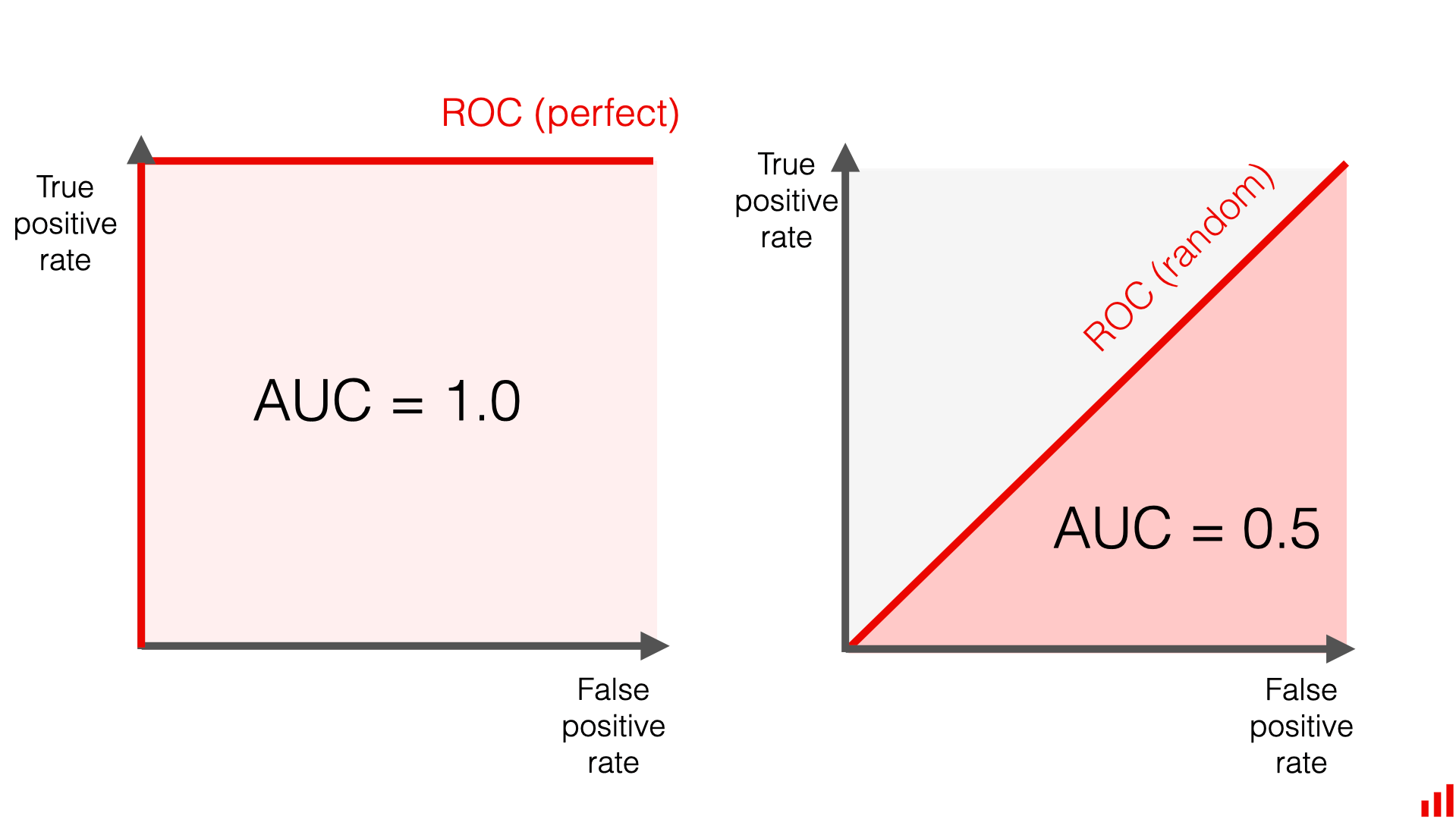
**Definition**:  
 Recall measures the model's ability to correctly identify all **actual positives** in the dataset.

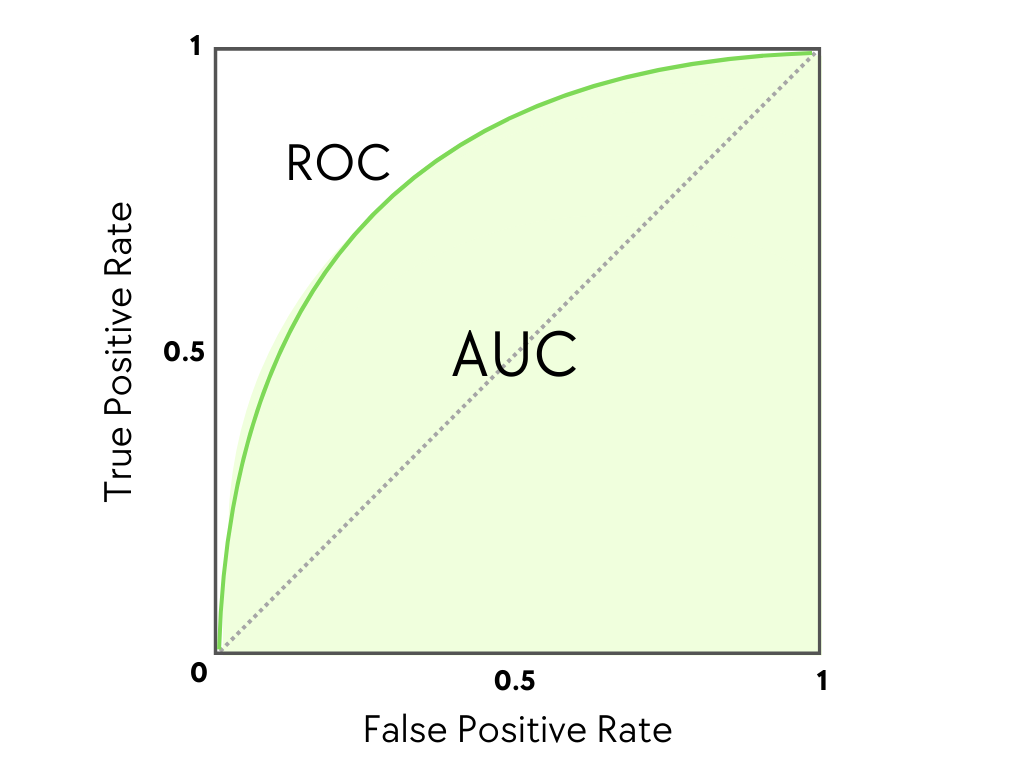
**Formula**:



### **iv) AUC-ROC (Area Under the Curve - Receiver Operating Characteristic)**

**Definition**:  
AUC-ROC is a **performance measurement** for classification problems at various threshold settings.





* **ROC Curve** plots:  
  + **True Positive Rate (Recall)** vs. **False Positive Rate (FPR = FP / (FP + TN))**
* **AUC (Area Under Curve)** gives a single value representing the model's ability to distinguish between classes:  
  + AUC = 1 → perfect classifier
  + AUC = 0.5 → no better than random guessing

**Interpretation**:  
 The **higher** the AUC, the **better** the model is at distinguishing between positive and negative classes.

Text Preprocessing Steps

**Text Preprocessing Steps**

**Introduction**

1. Text preprocessing is a critical step in Natural Language Processing (NLP) where r**aw text is transformed** into a format suitable for machine learning algorithms.
2. Since ML models work with numerical data, the unstructured text needs to be **cleaned and standardized** before encoding.
3. Preprocessing **improves the accuracy and performance** of NLP models by reducing noise and bringing uniformity to the data.

**1. Tokenization**

* **Definition**: Tokenization is the process of breaking a sentence or document into individual units called tokens (words, phrases, or characters).
* **Types**: Includes word-level, character-level, and subword (n-gram) tokenization.
* **Purpose**: Tokens help machines understand sentence structure and meaning.
* **Example**: Input: "Information Retrieval by Technical Publication" Tokens: ["Information", "Retrieval", "by", "Technical", "Publication"]

**2. POS Tagging (Part-of-Speech Tagging)**

* **Definition**: POS tagging is the process of **assigning a part of speech** (e.g., noun, verb, adjective) to each word in a sentence.
* **Purpose**: Enables deeper understanding of grammatical structure and context of words.
* **Use Case**: Crucial for lemmatization, syntactic parsing, named entity recognition, and question answering.
* **Example**:
  + Sentence:He is playing football.
  + POS tags: [('He', 'PRP'), ('is', 'VBZ'), ('playing', 'VBG'), ('football', 'NN')]

**3. Stemming**

* **Definition**: Stemming reduces words to their base or root form by removing suffixes.
* **Algorithm**: Uses rule-based approaches like the Porter stemmer.
* **Purpose**: Helps in grouping similar words with the same core meaning.
* **Example**: Words like "running", "runner" and "runs" are stemmed to "run".

**4. Stop Words Removal**

* **Definition**: Stop words are common words (e.g., “the”, “is”, “a”) that carry minimal semantic value.
* **Purpose**: Removing them reduces dataset size and training time.
* **Implementation**: A predefined stop word list is used for filtering.
* **Example**: Input: "The capital of India" After stopword removal: "capital India"

**5. Lemmatization**

* **Definition**: Lemmatization reduces words to their dictionary or base form (lemma) using vocabulary and morphological analysis.
* **Difference from Stemming**: Unlike stemming, it produces valid root words.
* **Use Case**: More accurate for tasks like search or translation.
* **Example**: Words like "am", "are", "is" → "be"; "better" → "good" (not possible with stemming)

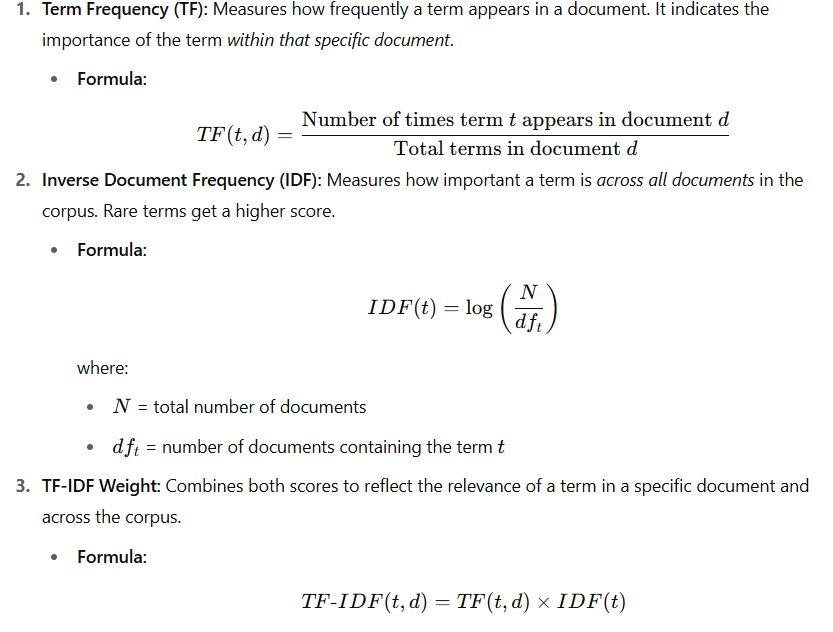
TF/IDF

**TF/IDF (Term Frequency-Inverse Document Frequency) in Text Analysis**

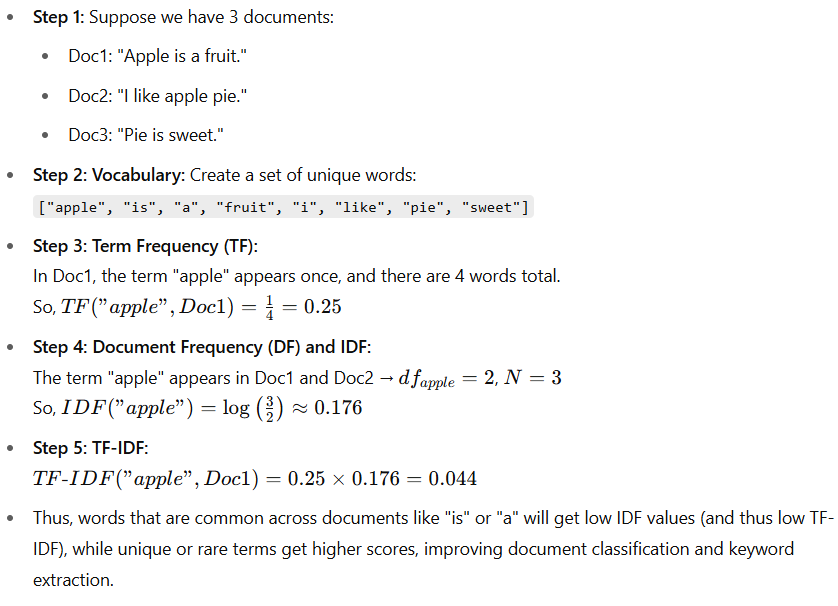
**Introduction:**

1. In text analysis and Natural Language Processing (NLP), it is important to determine which words in a document are more relevant or informative.
2. TF-IDF is a statistical measure used to evaluate the **importance of a word** in a document relative to a collection or corpus of documents.
3. It helps convert **raw textual data into a numerical representation** that highlights important keywords while reducing the weight of commonly used words.

**What is TF and IDF with Formula:**



**Usage in Text Analysis with Example:**



What is the Holdout Method, and How Does it Work

### **What is the Holdout Method, and How Does it Work?**

#### **Introduction:**

1. The **holdout method** is a simple and widely used technique for evaluating the performance of machine learning models.
2. It involves **splitting the original dataset** into separate subsets used for training and testing purposes.

### **Why is Holdout Method Required?**

* Machine learning models need to be evaluated on **unseen data** to check their **generalization ability**.
* Without splitting the data, a model may **overfit** to the training data and perform poorly on new data.

### **Process of Holdout Method:**

1. **Divide the dataset** into two (or three) parts — typically in ratios like 70/30, 80/20, or 60/40.
2. **Train the model** using the training set.
3. **Evaluate** the model using the test set.
4. Optionally, a **validation set** may be used to tune hyperparameters before final testing.

