Unit 4:

**What is Machine Learning?**

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that enables computers to learn patterns from data and make decisions or predictions without being explicitly programmed. Instead of using fixed rules, ML models improve their performance over time by analyzing large amounts of data.

**Why is Machine Learning Important?**

* Automates complex decision-making processes.
* Enhances accuracy and efficiency in data-driven tasks.
* Enables predictive analysis and pattern recognition.
* Powers modern AI applications like speech recognition, recommendation systems, and self-driving cars.

**Types of Machine Learning Algorithms**

Machine Learning algorithms can be categorized into the following types:

1. **Supervised Learning** – Uses labeled data to make predictions.
2. **Unsupervised Learning** – Works with unlabeled data to find hidden patterns.
3. **Semi-supervised Learning** – Uses both labeled and unlabeled data.
4. **Reinforcement Learning** – Learns from rewards and penalties to optimize actions.

**1. Supervised Learning**

Supervised learning involves training a model on a dataset that contains **input-output pairs** (labeled data). The model learns to map inputs to correct outputs and generalizes this knowledge to new data.

**Supervised Learning is divided into:**

1. **Classification:**
   * Predicts discrete categories (e.g., spam vs. not spam, disease detection).
   * **Common Algorithms:**
     + KNN(K nearest Neighbor)
     + Decision Trees
     + Random Forest
     + Support Vector Machines (SVM)
     + Neural Networks
2. **Regression:**
   * Predicts continuous values (e.g., house prices, temperature forecasting).
   * **Common Algorithms:**
     + Linear Regression
     + Polynomial Regression
     + Decision Tree Regression
     + Random Forest Regression
     + Support Vector Regression (SVR)

**2. Unsupervised Learning**

Unsupervised learning works with **unlabeled data** to discover patterns and relationships. The model learns structures from the input without explicit supervision.

**Unsupervised Learning is divided into:**

1. **Clustering:**
   * Groups similar data points together (e.g., customer segmentation, anomaly detection).
   * **Common Algorithms:**
     + K-Means Clustering
     + Hierarchical Clustering
     + DBSCAN
2. **Dimensionality Reduction:**
   * Reduces the number of features while preserving meaningful information (e.g., compressing high-dimensional images).
   * **Common Algorithms:**
     + Principal Component Analysis (PCA)
     + t-SNE (t-Distributed Stochastic Neighbor Embedding)

**3. Semi-Supervised Learning**

Semi-supervised learning combines both **labeled and unlabeled data** for training. It is useful when labeled data is limited but a large amount of unlabeled data is available.

**Example Use Cases:**

* Medical diagnosis (some labeled patient records, but a lot of unlabeled data).
* Web page classification (only some web pages are manually labeled).

**Common Algorithms:**

* Self-training
* Graph-based methods
* Semi-supervised Support Vector Machines

**4. Reinforcement Learning**

Reinforcement Learning (RL) is a trial-and-error learning approach where an **agent** interacts with an **environment** to maximize a reward.

**Key Components of Reinforcement Learning:**

* **Agent:** Learner or decision-maker.
* **Environment:** The system the agent interacts with.
* **Actions:** Choices available to the agent.
* **Rewards:** Feedback signal for actions taken.
* **Policy:** The strategy the agent follows.

**Common RL Algorithms:**

* Q-Learning
* Deep Q-Networks (DQN)
* Policy Gradient Methods

**Train-Test Split** – Why it is needed and how to implement it.

Train-test split is a technique used to evaluate machine learning models by dividing the dataset into:

* **Training set (80%)** – Used to train the model.
* **Test set (20%)** – Used to evaluate the model's performance on unseen data.

**Model Training** – Training a machine learning model using a dataset.

**High Bias (Underfitting):** If k is **too large**, the model oversimplifies patterns.

**High Variance (Overfitting):** If k is **too small**, the model memorizes training data.

**Bias and Variance** – Understanding their impact on model performance.

| **Model** | **Train Accuracy** | **Test Accuracy** |
| --- | --- | --- |

| (Underfitting) | Low | Low |
| --- | --- | --- |

| (Overfitting) | High | Low |
| --- | --- | --- |

**4. Overfitting vs. Underfitting**

**Overfitting**

Occurs when a model memorizes the training data instead of learning patterns. **Symptoms**:

* High accuracy on training data but poor performance on test data.

**Underfitting**

Occurs when a model is too simple and cannot learn from training data. **Symptoms**:

* Poor accuracy on both training and test data.

**Model Evaluation**

Evaluating a machine learning model is essential to determine how well it **generalizes** to unseen data. Evaluating a machine learning model is essential to understand its **performance** and **generalization ability**.The key aspects of model evaluation in **classification problems** include:

1. **Performance Metrics** – Accuracy, Precision, Recall, F1-score,
2. **Confusion Matrix** – To analyze classification results.

**Accuracy**

Accuracy is the **ratio of correctly predicted observations to the total observations**. It is the simplest evaluation metric.

Accuracy=CorrectPredictions /TotalPredictions

or

**2. Confusion Matrix**

**Definition:**  
A **confusion matrix** is a table that describes the **performance of a classification model** by comparing **actual labels vs. predicted labels**. In classification problems, model performance is evaluated using **True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN)**.

**TP (True Positive)** → Model **correctly** predicted **Positive**.

* Example: A cancer detection model correctly detects cancer.

**TN (True Negative)** → Model **correctly** predicted **Negative**.

* Example: A spam filter correctly classifies an email as **not spam**.

**FP (False Positive)** → Model **incorrectly** predicted **Positive**.

* Example: A spam filter wrongly marks an important email as **spam**.

**FN (False Negative)** → Model **incorrectly** predicted **Negative**.

**3. Precision, Recall, and F1-Score**

**(a) Precision**

Precision tells us how many of the predicted **positive cases were actually correct**.

**Precision=TP/(TP+FP)**

**(b) Recall (Sensitivity / True Positive Rate)**

Recall measures **how many actual positive cases were correctly identified** by the model.

**Recall=TP/(TP+FN)**

**(c) F1-Score**

The F1-score is the **harmonic mean of Precision and Recall**, balancing both metrics.

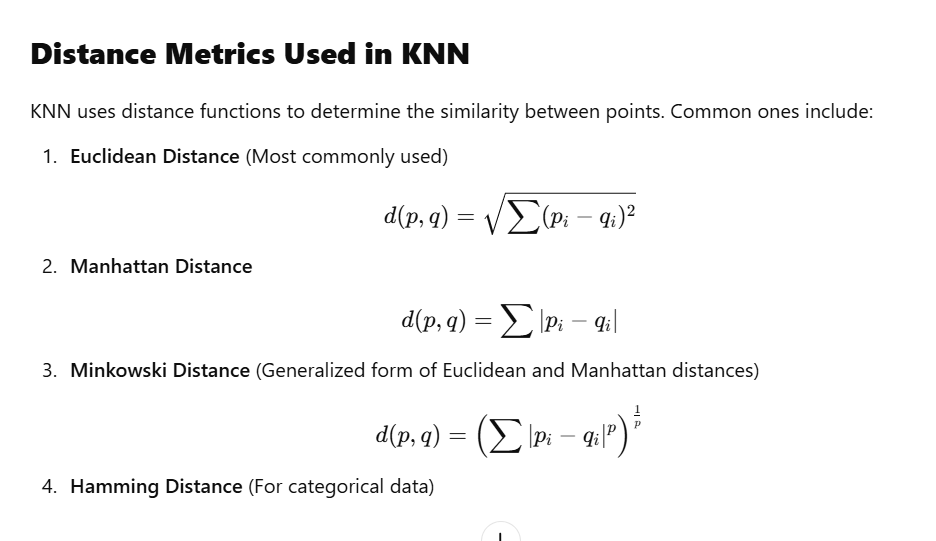
**F1=2×Precision×Recall/(Precision+Recall)**

**K-Nearest Neighbors (KNN) Algorithm**

K-Nearest Neighbors (KNN) is a **supervised machine learning algorithm** used for both **classification** and **regression** tasks. It is a **non-parametric, instance-based** algorithm that makes predictions based on the similarity between data points.

How KNN Works?

1. **Choose the number of neighbors (K).**
2. **Measure the distance** between the new data point and all training samples using a distance metric (e.g., Euclidean distance).
3. **Find the K nearest neighbors** (smallest distances).
4. **For classification:**
   * Assign the most common class (majority voting) among the K nearest neighbors.
5. **For regression:**
   * Predict the average (or weighted average) of the neighbors' values.



* When you want to classify a data point into a category (like spam or not spam), the K-NN algorithm looks at the **K closest points** in the dataset. These closest points are called neighbors. The algorithm then looks at which category the neighbors belong to and picks the one that appears the most. This is called **majority voting**.
* In regression, the algorithm still looks for the **K closest points**. But instead of voting for a class in classification, it takes the **average** of the values of those K neighbors. This average is the predicted value for the new point for the algorithm.

**Applications**

* **Recommendation Systems**: Many recommendation systems, such as those used by Netflix or Amazon, rely on KNN to suggest products or content. KNN observes at user behavior and finds similar users. If user A and user B have similar preferences, KNN might recommend movies that user A liked to user B.
* **Spam Detection**: KNN is widely used in filtering spam emails. By comparing the features of a new email with those of previously labeled spam and non-spam emails, KNN can predict whether a new email is spam or not.
* **Customer Segmentation**: In marketing firms, KNN is used to segment customers based on their purchasing behavior . By comparing new customers to existing customers, KNN can easily group customers into segments with similar choices and preferences. This helps businesses target the right customers with right products or advertisements.
* **Speech Recognition**: KNN is often used in speech recognition systems to transcribe spoken words into text. The algorithm compares the features of the spoken input with those of known speech patterns. It then predicts the most likely word or command based on the closest matches.

**Evaluation of Model :Regression**

Evaluation Measures in Regression problems

**MAE** → If you need an easy-to-interpret absolute error measure.

**MSE/RMSE** → If large errors should be penalized more.

**MAPE** → When percentage errors are important.

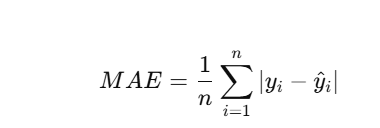
**R² Score** → When assessing how well the model explains variance.

**1. Error-Based Metrics**

These metrics measure how far the predicted values are from the actual values.

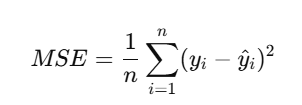
**(i) Mean Absolute Error (MAE)**

* Measures the average absolute difference between actual and predicted values.



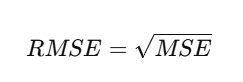
**(ii) Mean Squared Error (MSE)**

* Measures the **average squared** difference between actual and predicted values.



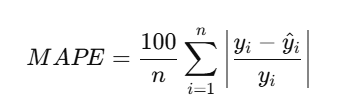
**(iii) Root Mean Squared Error (RMSE)**

* Square root of MSE, gives error in the same unit as the target variable.



**(iv) Mean Absolute Percentage Error (MAPE)**

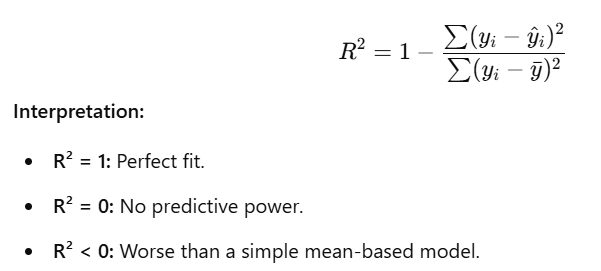
* Measures error as a percentage of actual values.



2.Goodness-of-Fit Metrics

**(v) R² Score (Coefficient of Determination)**

* Measures how well the model explains the variance in the data.



**1. What is Linear Regression?**

Linear Regression is a **supervised learning algorithm** used for **predicting a continuous value** based on input features. It establishes a relationship between the **dependent variable (Y)** and one or more **independent variables (X)** using a linear equation. statistical method

**Equation of Linear Regression**

For a **simple linear regression** (one independent variable):

Y==mX+c

where:

* Y = Predicted output (dependent variable)
* X = Input feature (independent variable)
* m= Slope (coefficient of X)
* c= Intercept (bias)

**Types of Linear Regression**

**When there is only one independent feature it is known as Simple Linear Regression or**[**Univariate Linear Regression**](https://www.geeksforgeeks.org/univariate-linear-regression-in-python/)**and when there are more than one feature it is known as Multiple Linear Regression or**[**Multivariate Regression**](https://www.geeksforgeeks.org/multivariate-regression/)**.**

**Unsupervised Learning**

**K-Means Clustering:**

**K-Means is an unsupervised machine learning algorithm used for clustering data into K distinct groups. It works by partitioning a dataset into K clusters, where each data point belongs to the nearest cluster centroid.**

**Advantages of K-Means Clustering**

1. **Simple & Efficient: Easy to implement and computationally efficient for large datasets.**
2. **Scalability: Can handle large datasets efficiently compared to hierarchical clustering.**
3. **Fast Convergence: Converges quickly to a solution, especially with the K-Means++ initialization.**
4. **Handles Different Data Types: Works well with numerical data.**
5. **Interpretable Results: Provides well-defined cluster assignments.**

**Working of K-Means Algorithm**

1. **Choose K (Number of Clusters)**
   * **Decide how many clusters you want to divide your data into.**
2. **Initialize Centroids**
   * **Randomly select K initial points as cluster centroids.**
3. **Assign Data Points to the Nearest Centroid**
   * **Compute the Euclidean distance between each data point and centroids.**
   * **Assign each data point to the closest centroid.**
4. **Recalculate Centroids**
   * **Compute the new centroid by taking the mean of all points in each cluster.**
5. **Repeat Until Convergence**
   * **Continue steps 3 and 4 until centroids no longer change or a predefined number of iterations is reached.**

**These metrics assess clustering quality without ground truth labels by analyzing intra-cluster and inter-cluster distances.**

**(a) Silhouette Score**

* **Measures how well each data point fits into its assigned cluster.**
* **Ranges from -1 to 1:**
  + **1 → Well-clustered**
  + **0 → On the boundary**
  + **-1 → Poor clustering (incorrect assignment**

**Naive bayes Algorithm**

Naïve Bayes is a probabilistic classifier based on Bayes’ Theorem, assuming that features are independent (hence, "naïve"). It is widely used for text classification, including spam detection.

Bayes theorem

P(A/B)=P(B/A).P(A)/P(B)

**Types of Bayes algorithm**

Gaussian Naive Bayes: continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution.

Multinomial Naive Bayes: It is used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

Bernoulli Naive Bayes deals with binary features, where each feature indicates whether a word appears or not in a document.

**Singular Value Decomposition (SVD)**

Singular Value Decomposition (SVD) is a **matrix factorization** technique used in **dimensionality reduction, noise reduction, and feature extraction**. It decomposes a matrix into three matrices:

Eg with rows are people, and columns are products. The numbers in the table show how much each person likes each product. SVD helps you split that table into three parts:

* **U**: This part tells you about the people (like their general preferences).
* **Σ**: This part shows how important each factor is (how much each rating matters).
* **Vᵀ**: This part tells you about the products (how similar they are to each other)

Mathematically, the SVD of a matrix A (of size m×n*m*×*n*) is represented as:

A=UΣVT

The SVD decomposition function simplifies complex data by breaking it into three smaller parts. This helps uncover hidden patterns and relationships, making it easier to analyse and work with large datasets.