AILM

CO-3

1). Introduction to Machine Learning

Machine Learning (ML) is when we teach computers to learn from data instead of giving them strict rules. It's like showing examples to a computer and letting it figure things out. For example, ML helps your phone suggest the next word while texting or sort your photos by faces.

Types of Machine Learning

1. Supervised Learning

- The computer learns by studying examples with the right answers.
- **Example:** Teach the computer to recognize animals by showing pictures of cats and dogs with labels like "cat" or "dog." Then it can guess correctly for new pictures.

2. Unsupervised Learning

- The computer looks at data with no labels and finds patterns or groups on its own.
- **Example:** It groups customers into similar types based on their buying habits without being told who they are.

3. Reinforcement Learning

- The computer learns by trial and error, getting rewards for good actions and penalties for bad ones.
- **Example:** A robot learns to walk by getting points for each successful step.

---

2). Introduction to Data Preprocessing

Data preprocessing is the process of preparing raw data for machine learning models. Since raw data can be messy, preprocessing helps clean and format it so models can learn effectively. It's a critical step to improve accuracy and efficiency in machine learning.

Steps in Data Preprocessing

1. Data Cleaning

- Removing or fixing incorrect, incomplete, or duplicate data.
- **Example:** Filling in missing values, fixing typos, or removing rows with errors.

2. Data Splitting

- Dividing the data into:
  - **Training Set:** Used to teach the model.
  - **Validation Set:** Used to tune the model.
  - **Testing Set:** Used to check how well the model works.
- **Example:** Splitting 80% of data for training and 20% for testing.

3. Data Normalization

- Scaling data so that all features have similar ranges or distributions.
- **Example:** Converting values like "height in cm" and "weight in kg" to a scale between 0 and 1 for easier comparison.

4. Data Batching

- Dividing large datasets into smaller groups (batches) to train the model in chunks.
- **Why:** It helps with memory management and speeds up training.

5. Data Shuffling

- Randomly reordering the data before training.
- **Why:** Prevents the model from learning patterns based on the order of data instead of actual features.

---

4).Overfitting and Underfitting

Overfitting

- The model learns too much detail from the training data and performs poorly on new data.
- **Example:** A model perfectly predicts training data but fails on test data.

Underfitting

- The model doesn't learn enough and performs poorly on both training and new data.
- **Example:** A model that doesn't recognize patterns in the data at all.

---

5).Confusion Matrix - Simple Explanation

A **confusion matrix** is a table that helps check how well a model works. It shows:

- **Correct Predictions:** How many times the model got it right.
- **Wrong Predictions:** How many times the model got it wrong.

It's useful for both:

- **Binary Classification:** Two options (e.g., Yes/No).
- **Multi-Class Classification:** Many options (e.g., Cat/Dog/Rabbit).

For binary problems, the table is 2x2:

- Rows: What's true (actual values).
- Columns: What the model guessed (predicted values).

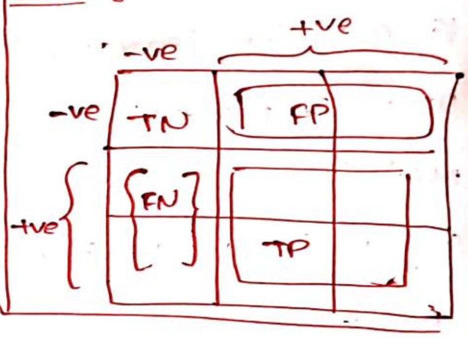
Confusion Matrix :

Confusion Matrix

Case 1:

	-ve	+ve
-ve	TN	FP
+ve	FN	TP

Case 2:



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

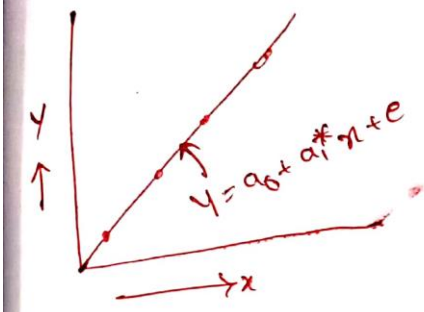
$$\text{Precision} = \frac{TP}{TP + FP}$$

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

$$\text{f-Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

linear Regression

examples



x_i	y_i	xy	x^2
1.	1.2		
2	1.8		
3	2.6		
4	3.2		
5	3.8		

Single
* linear regression equation

$$\boxed{y = a_0 + a_1 \times x + e}$$

where,

$$a_1 = \frac{(\overline{xy}) - (\bar{x})(\bar{y})}{\overline{x^2} - \bar{x}^2}$$

$$a_0 = \bar{y} - a_1 \times \bar{x}$$

* Multi L.R.

$$\boxed{y = a + bx}$$

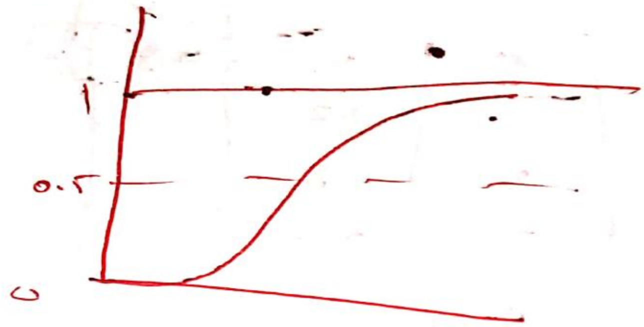
$$b = \frac{\Sigma (x - \bar{x}) \cdot (y - \bar{y})}{\Sigma \cdot (x - \bar{x})^2}$$

b is slope

a is intercept.

$$\therefore a = \bar{y} - b\bar{x}$$

logistic Regression



$$S(z) = \frac{1}{1+e^{-z}}$$

$$z = \beta_0 + \beta_1 \cdot x$$

Decision Tree.

$$\text{Entropy}(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

$$\boxed{\begin{aligned} &S(+\text{ve examples}, -\text{ve examples}) \\ &= -P_+ \log_2 P_+ - P_- \log_2 P_- \end{aligned}}$$

$$\text{Gain}(\text{Sunny}, \text{wind}) = \text{Entropy}(S) - \sum_{v \in (S, W)} \frac{|S_v|}{|S|} \text{Entropy}(S_v$$

KNN

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

CO-4

13. Introduction to Ensemble Methods

Ensemble methods combine multiple models to improve performance and reduce overfitting. These methods are divided into two main types:

Bagging (Bootstrap Aggregating):

- **Random Forest:**
  - **Definition:** An ensemble of decision trees created by training each tree on a random subset of data (with replacement) and averaging their predictions (regression) or using majority voting (classification).
  - **Key Features:**
    - Reduces overfitting by averaging multiple models.
    - Handles missing data well.
    - Works well for both regression and classification tasks.
  - **Steps:**

1. Randomly sample data with replacement to create multiple subsets.
2. Build a decision tree for each subset.
3. Aggregate results from all trees for the final prediction.

Boosting:

Boosting builds models sequentially, where each model attempts to correct the errors of the previous one. Popular techniques include:

- **XGBoost:**
  - Uses gradient boosting algorithms with speed and performance optimizations.
  - Handles missing data and is highly regularized to prevent overfitting.

- Widely used in competitions like Kaggle for its accuracy and efficiency.

- **AdaBoost (Adaptive Boosting):**
    - Assigns higher weights to misclassified data points in subsequent models.
    - Combines weak classifiers into a strong classifier.
    - Typically uses decision stumps (single-split decision trees) as weak learners.

---

14. Unsupervised Learning Algorithms

Unsupervised learning identifies patterns in data without labeled outputs.

K-Means Clustering:

- **Definition:** Partitions data into kkk clusters based on the similarity of data points.
- **Steps:**
    1. Initialize kkk centroids randomly.
    2. Assign each data point to the nearest centroid.
    3. Recalculate centroids based on the mean of assigned points.
    4. Repeat until centroids stabilize.
- **Applications:** Customer segmentation, market research, image compression.

Hierarchical Clustering:

- **Definition:** Builds a hierarchy of clusters by either merging smaller clusters (agglomerative) or splitting larger clusters (divisive).
- **Types:**
    - **Agglomerative:** Starts with each data point as a cluster and merges them iteratively.
    - **Divisive:** Starts with one cluster and splits it iteratively.

- **Applications:** Social network analysis, bioinformatics, anomaly detection.

---

15. Artificial Neural Networks (ANN)

ANNs are inspired by biological neural networks and are fundamental to deep learning.

Introduction to ANN:

- **Weights and Bias:**
    - Weights determine the strength of connections between neurons.
    - Bias adjusts the activation function's threshold, improving flexibility.
- **Bias vs Variance:**
    - **Bias:** Error from oversimplified assumptions (underfitting).
    - **Variance:** Error from sensitivity to small fluctuations in the training set (overfitting).
- **McCulloch Pitts Model:**
    - The first computational model of a neuron.
    - Uses binary inputs/outputs and applies a threshold function for decision-making.
- **Perceptron:**
    - A simple model for supervised binary classification.
    - Learns weights using the perceptron learning rule.
- **Applications of ANN:**
    - Image and speech recognition.
    - Financial forecasting.
    - Medical diagnosis.
    - Recommendation systems.

Types of ANN:

- **Single-Layer Perceptron:** Processes linearly separable data using a single layer of neurons.
- **Multi-Layer Perceptron (MLP):**
    - Contains input, hidden, and output layers.
    - Uses activation functions and backpropagation.
- **Feedforward and Backpropagation Networks:**
    - **Feedforward:** Data flows from input to output without loops.
    - **Backpropagation:** Optimizes weights using gradient descent.
- **Recurrent Neural Networks (RNN):**
    - Handles sequential data by maintaining a memory of previous inputs.
    - Applications: Language models, time series prediction.

Basics of ANN:

- **Structure:** Layers of interconnected neurons.
- **Functionality:** Processes inputs through weighted connections and activation functions.
- **Learning in ANN:** Adjusts weights using optimization algorithms.

ANN Techniques:

- **Activation Functions:**
    - **Sigmoid:** Outputs in the range [0, 1].
    - **Tanh:** Outputs in the range [-1, 1].
    - **ReLU:** Outputs max(0,x)\text{max}(0, x)max(0,x); widely used in hidden layers.
- **Error Computation (Loss Functions):**
    - **Mean Squared Error (MSE):** For regression tasks.
    - **Cross-Entropy Loss:** For classification tasks.
- **Error Optimization:**

    - Optimizes weights using techniques like gradient descent, RMSProp, and Adam.
- **Prediction Using ANN:**
    - Feeds test data into the trained network to predict outcomes.