CONTINUOUS ASSESSMENT TEST – II

Second Year /Third Semester

2311CSC303J – Fundamentals of Artificial Intelligence and Machine Learning

(Common to AIADS, CSE, IT, CSE (AIML), CSE (CS), CSBS, CSD)

Unit – III Introduction to Machine Learning

<table>
<tr><th>Q.No</th><th>Questions</th></tr>
<tr><td colspan="2">Part A</td></tr>
<tr><td>1</td><td>State any two real-life applications of supervised learning
Email Spam Detection:
Supervised learning algorithms are trained on labeled emails (spam or not spam) to classify new incoming emails automatically.
Medical Diagnosis:
Models are trained on patient data with known diagnoses to predict diseases or health conditions for new patients based on their medical parameters.</td></tr>
<tr><td>2</td><td>Why is training data important in supervised learning?
Training data is important because it provides labeled examples that help the model learn the relationship between input features and output labels. The quality and quantity of training data directly affect the model’s accuracy and performance.</td></tr>
<tr><td>3</td><td>List two key differences between classification and regression
<table><tr><th>Basis</th><th>Classification</th><th>Regression</th></tr><tr><td>Output Type</td><td>Predicts categorical values (e.g., Yes/No, Spam/Not Spam)</td><td>Predicts continuous values (e.g., Salary, Temperature)</td></tr><tr><td>Evaluation Metric</td><td>Measured using accuracy, precision, or F1-score</td><td>Measured using mean squared error (MSE) or R^2 score</td></tr></table></td></tr>
<tr><td>4</td><td>Name any one algorithm used for classification and one for regression</td></tr>
</table>

	<table><tr><th>Task</th><th>Common Algorithms</th></tr><tr><td>Classification</td><td>Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Neural Networks</td></tr><tr><td>Regression</td><td>Linear Regression, Polynomial Regression, Decision Tree Regressor, Random Forest Regressor, Support Vector Regressor (SVR), Ridge Regression, Lasso Regression, Neural Networks</td></tr></table>
5	Explain how the output of classification differs from that of regression. In classification, the output is categorical, meaning it predicts discrete class labels such as yes/no or spam/not spam. In regression, the output is continuous, meaning it predicts numeric values such as salary, temperature, or price.
6	Define the term overfitting in machine learning. Overfitting occurs when a machine learning model learns the training data too well, including its noise and errors. As a result, it performs well on training data but poorly on new, unseen data.
7	What do you understand by generalization in machine learning? Why is it important? Generalization is the ability of a machine learning model to perform well on new, unseen data, not just the training data.
8	Differentiate between underfitting and overfitting with respect to model performance <table><tr><th>Basis</th><th>Underfitting</th><th>Overfitting</th></tr><tr><td>Definition</td><td>The model is too simple and fails to capture the underlying patterns in the data</td><td>The model is too complex and learns noise along with the patterns in the training data</td></tr><tr><td>Performance on Training Data</td><td>Poor</td><td>Very good</td></tr><tr><td>Performance on Test Data</td><td>Poor</td><td>Poor</td></tr><tr><td>Cause</td><td>Insufficient model complexity or features</td><td>Excessive model complexity or too much training</td></tr></table>
9	List any two supervised machine learning algorithms with its advantages and disadvantages. <table><tr><th>Algorithm</th><th>Advantages</th><th>Disadvantages</th></tr><tr><td>Decision Tree</td><td>Easy to understand and interpret; handles both numerical and categorical data</td><td>Prone to overfitting; can be unstable with small variations in data</td></tr><tr><td>Linear Regression</td><td>Simple to implement; works well for linearly related data; fast computation</td><td>Cannot capture complex non-linear relationships; sensitive to outliers</td></tr></table>

10	What is the main difference between the Decision Tree and K-Nearest neighbours (KNN) algorithms. The main difference is that Decision Tree is a model-based algorithm that learns decision rules from the training data to make predictions, while K-Nearest Neighbours (KNN) is a instance-based algorithm that makes predictions by comparing new data points to the closest training examples without building an explicit model.

Q. No	Questions
	Part B
1.	Discuss in detail the mathematical and algorithmic indicators of overfitting in supervised learning. With examples, evaluate how overfitting affects model performance on unseen data. Analyse advanced generalization techniques with examples. Overfitting is a common problem in supervised learning where a model performs very well on the training data but fails to perform accurately on unseen or test data. This happens when the model learns not only the actual patterns but also the noise and errors present in the training dataset. For example, a model predicting student marks may learn the exact data points of the training set but fail when a new student's data is given. 1. **Error Difference:** Overfitting is seen when the **training error (E_train)** is very low and the **testing error (E_test)** is very high. $$E_{train} \ll E_{test}$$ Example: A model gets 2% error on training data but 25% error on testing data — clear sign of overfitting. 2. **Bias-Variance Trade-off:** The total error in a model can be expressed as: $$Total\ Error = Bias^2 + Variance + Irreducible\ Error$$ • Overfitting occurs when the **variance** is high and **bias** is low. • The model becomes too sensitive to small changes in the data.

Algorithmic Indicators of Overfitting

1. **Model Complexity:**
   When the model is too complex, like a deep decision tree or a high-degree polynomial, it starts to memorize training data.
2. **Learning Curve Pattern:**
   - Training accuracy keeps increasing.\
   - Validation or test accuracy stops improving or even decreases.
3. **Small Dataset:**
   With limited data, the model may not see enough examples and starts memorizing instead of learning general patterns.

Dataset	Model Performance
Training Data	Very high accuracy (model memorizes patterns)
Testing Data	Low accuracy (fails to generalize)

Generalization is the ability of a model to perform well on new, unseen data.
A model that generalizes well is not too simple (underfitting) and not too complex (overfitting).

Techniques to Improve Generalization (Reduce Overfitting)

Technique	Description	Example / Use Case
Regularization (L1 & L2)	Adds a penalty term to reduce large weights and prevent complex models.	Ridge (L2) and Lasso (L1) Regression
Dropout	Randomly turns off some neurons during training to prevent dependency on specific nodes.	Used in Deep Neural Networks
Cross-Validation	Splits data into multiple folds to test performance across different sets.	K-Fold Cross-Validation
Early Stopping	Stops training when validation loss starts increasing, even if training loss is still reducing.	Used in deep learning models

2. Explain how underfitting can be detected through model evaluation metrics and learning curves. Using appropriate examples, examine the impact of underfitting in both regression and classification tasks.

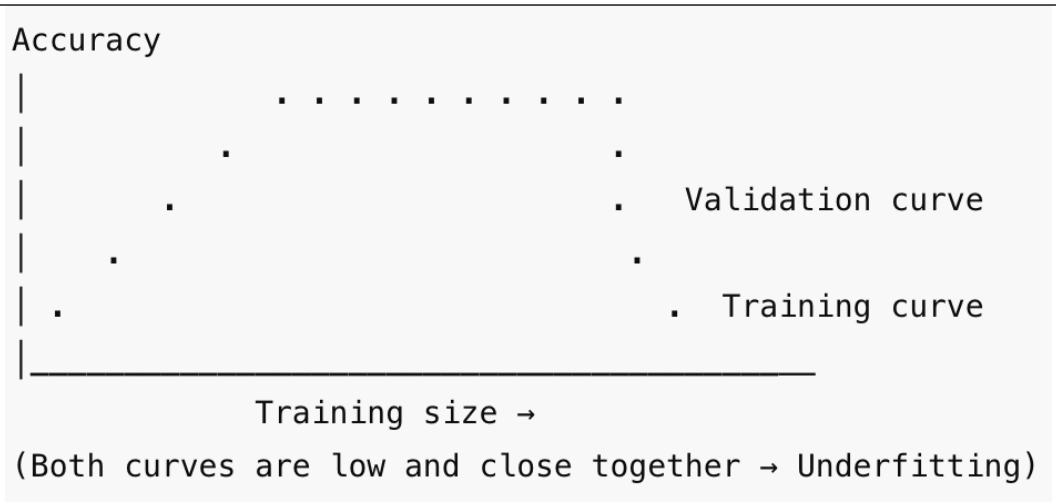
Underfitting occurs when a machine learning model is too simple to learn the underlying patterns of the training data. It results in poor performance on both training and testing data, as the model fails to capture important relationships in the dataset.

1. Detecting Underfitting Using Model Evaluation Metrics

1. High Training Error and High Testing Error:
   Underfitting can be detected when both training and testing errors are large.
   - The model performs poorly on the training data itself, indicating that it has not learned enough.
   - Example metrics:
     - Regression: High Mean Squared Error (MSE), low R^2 value.
     - Classification: Low accuracy, high misclassification rate.
2. Example:
3. Low Accuracy Across Datasets:
   If the model's accuracy is low for both training and validation data, it is a clear sign of underfitting.

2. Learning curves plot training and validation performance against training size or epochs.

- In underfitting:
  - Training accuracy is low.
  - Validation accuracy is almost the same as training accuracy (both poor).
  - Increasing training data or training time does not improve performance.



3. Example in Regression

Example:

We try to predict student marks based only on study hours, but in reality, marks also depend on attendance, assignments, and previous performance.

The model will give poor predictions because it doesn't capture all important factors.

Effect:

- Training error is high.
- Test error is also high.
- The line of regression doesn't fit the points properly.

4. Example in Classification

Effect:

- Many misclassifications.

	● Both training and testing accuracy are low.
3.	Design a supervised learning pipeline incorporating preprocessing steps, choice of algorithm(s), and model evaluation techniques. Explain how you would tune the hyperparameters and optimize the model performance while preventing overfitting. Justify your decisions at each step. A supervised learning pipeline is a structured workflow that takes raw data, processes it, trains a model, and evaluates its performance. **1. Problem Understanding and Data Collection** ● The first step is to define the problem clearly. ○ Classification Example: Predict whether a student passes or fails. ○ Regression Example: Predict the price of a house. ● Understanding the problem helps to select the appropriate algorithm, evaluation metrics, and preprocessing steps. **2. Data Preprocessing** Data preprocessing is crucial because real-world data is often noisy, incomplete, or inconsistent. Preprocessing steps include: 1. Handling Missing Values: ○ Replace missing values with mean/median for numerical features or mode for categorical features. ○ Prevents the model from failing or learning incorrect patterns. 2. Encoding Categorical Features: ○ Convert non-numeric data to numeric using One-Hot Encoding or Label Encoding. ○ Example: Gender ("Male/Female") → 0/1. 3. Feature Scaling: ○ Standardize or normalize features so that algorithms like KNN, SVM, and Neural Networks perform better. 4. Train-Test Split:

- Split the dataset into training (70–80%) and testing (20–30%) sets.
- Ensures that the model is evaluated on unseen data.

5. Optional – Feature Selection or Dimensionality Reduction:
   - Remove irrelevant or redundant features.
   - Apply PCA if features are highly correlated to reduce complexity and prevent overfitting.

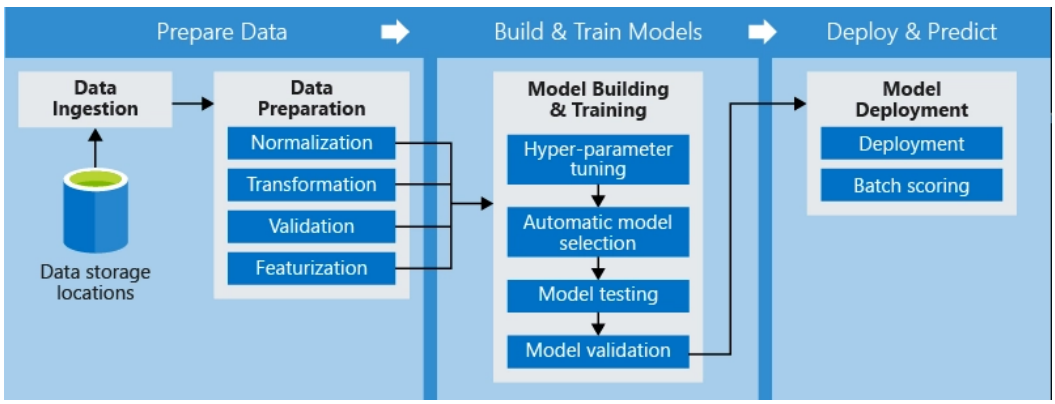
3. Choice of Algorithm

Selection depends on:

- Type of task: Classification or Regression
- Data size, feature types, and complexity
- Need for interpretability

4. Model Training

- Train the model on preprocessed training data.
- Use cross-validation to monitor performance on validation sets and ensure that the model generalizes well.



5. Model Evaluation

- Classification Metrics: Accuracy, Precision, Recall, F1-Score, ROC-AUC
- Regression Metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R^2 score

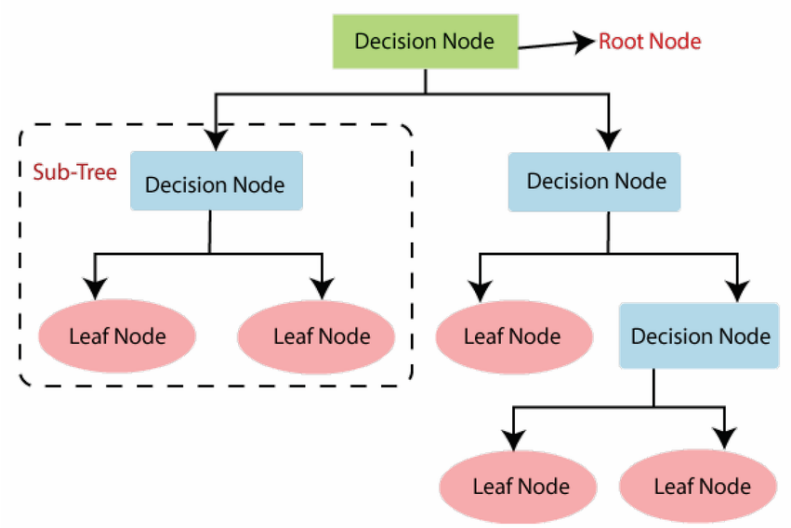
6. Hyperparameter Tuning

- Use Grid Search, Random Search, or Bayesian Optimization to find the best hyperparameters.

	7. Preventing Overfitting Overfitting occurs when the model memorizes training data instead of learning general patterns. Techniques to prevent it include: 1. Regularization: L1/L2 penalties restrict model complexity. 2. Pruning: Limit tree depth or minimum samples in leaf nodes (for Decision Trees). 3. Dropout: Randomly deactivate neurons in neural networks during training. 4. Cross-Validation: Ensures model generalizes across data splits. 5. Early Stopping: Stop training when validation error starts to increase. **8. Model Optimization** ● Feature Engineering: Create new features that capture important relationships. ● Ensemble Methods: Combine multiple models (Random Forest, Gradient Boosting) to reduce variance and improve stability. ● Iterative Refinement: Continuously evaluate model performance, adjust features and hyperparameters, and retrain until satisfactory results are achieved.
4.	Analyse and describe in detail about at least three supervised machine learning algorithms (e.g., Decision Trees, Support Vector Machines, and k-Nearest Neighbours) in terms of their underlying working principles, computational complexity, and suitability for different types of datasets. Supervised machine learning algorithms are designed to learn a mapping from input features to known output labels. Three commonly used algorithms are Decision Trees (DT), Support Vector Machines (SVM), and k-Nearest Neighbors (KNN). 1. Decision Trees (DT) ● A Decision Tree is a tree-structured model that recursively splits the dataset based on feature values to create a hierarchical structure of decisions. ● Each internal node represents a feature test (e.g., “Is Age > 18?”), branches represent outcomes of the test, and leaf nodes represent predicted class labels or continuous output values. ● **Splitting Criteria:**

- Classification: Gini Index, Entropy (Information Gain)
- Regression: Mean Squared Error (MSE) or Variance Reduction

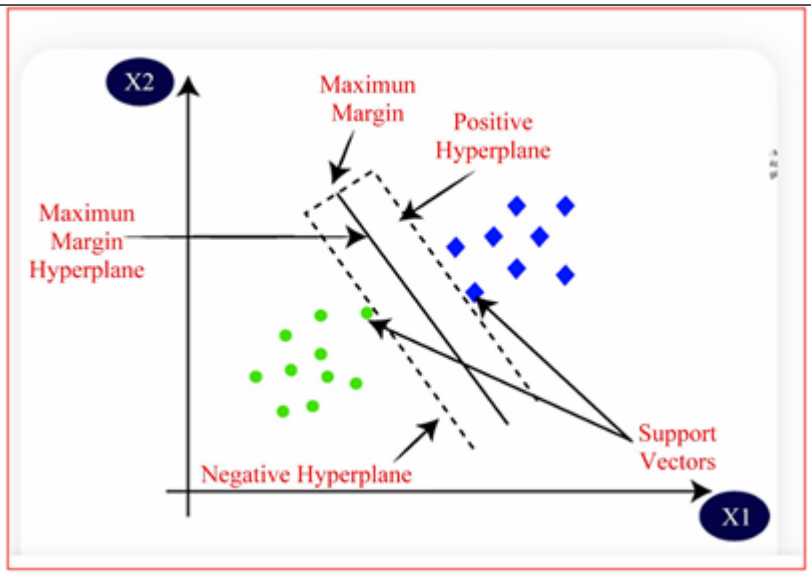
- The tree grows until stopping criteria are met (e.g., max depth, minimum samples per leaf).



Overview of Decision Tree

Support Vector Machines (SVM)

- SVM aims to find a hyperplane that best separates classes in a high-dimensional space.
- Key Concepts:
  - Margin: Distance between the hyperplane and the closest points (support vectors) of each class.
  - Optimal Hyperplane: Maximizes the margin for better generalization.
- Kernel Trick: Allows SVM to handle non-linear separable data by mapping it into higher-dimensional space (common kernels: Linear, Polynomial, RBF).

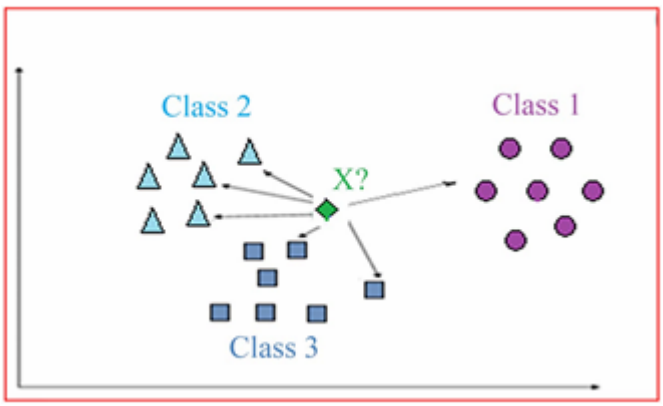


Overview of SVM

k-Nearest Neighbors (KNN)

Working Principle

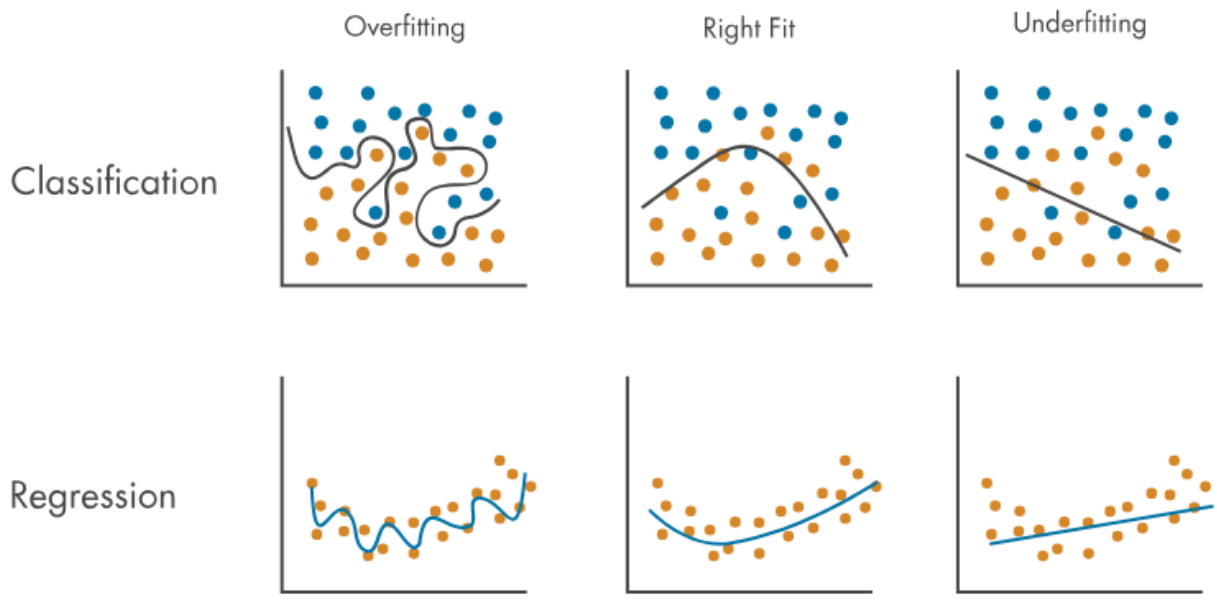
- KNN is a lazy learning algorithm (does not build a model during training).
- Prediction is based on the majority class (classification) or average value (regression) of the k closest training samples in the feature space.
- Distance metrics commonly used: Euclidean, Manhattan, or Minkowski.



Overview of KNN

Algorithm	Working Principle	Training Complexity	Prediction Complexity	Strengths	Weaknesses	Best For
Decision Tree	Recursive feature-based splits	$O(n \cdot m \cdot \log n)$	O(depth)	Interpretable, handles categorical & numerical data	Overfitting, sensitive to noise	Small-medium datasets, interpretable tasks
SVM	Find hyperplane with max margin	$O(n^2 \text{ to } n^3)$	$O(s \cdot m)$	High-dimensional data, effective margin separation	Computationally expensive, sensitive to kernel	Text, image, bioinformatic datasets
KNN	Assign label based on nearest neighbors	O(1)	$O(n \cdot m)$ per query	Simple, non-parametric, captures complex boundaries	Slow on large datasets, sensitive to scaling	Small-medium datasets, non-linear classification/regression

Q.No	Questions
	Part C
1.	A retail company has two models: one achieves 98% training accuracy but only 60% test accuracy, while another has low accuracy in both training and test sets. Design an improved modelling strategy to address both overfitting and underfitting scenarios. Propose suitable techniques (e.g., data augmentation, regularization, hyperparameter tuning, ensemble methods) and create a structured plan showing how you would implement these solutions. Justify how your proposed approach ensures better model performance on unseen data. This scenario involves two common problems in supervised learning: overfitting and underfitting. • Model 1: 98% training accuracy, 60% test accuracy → overfitting • Model 2: Low accuracy on both training and test sets → underfitting



Model	Training Accuracy	Test Accuracy	Problem	Cause
Model 1	98%	60%	Overfitting	Model memorizes training data; complex model or insufficient regularization
Model 2	Low	Low	Underfitting	Model too simple, not enough features, or insufficient training

Step-by-Step Plan

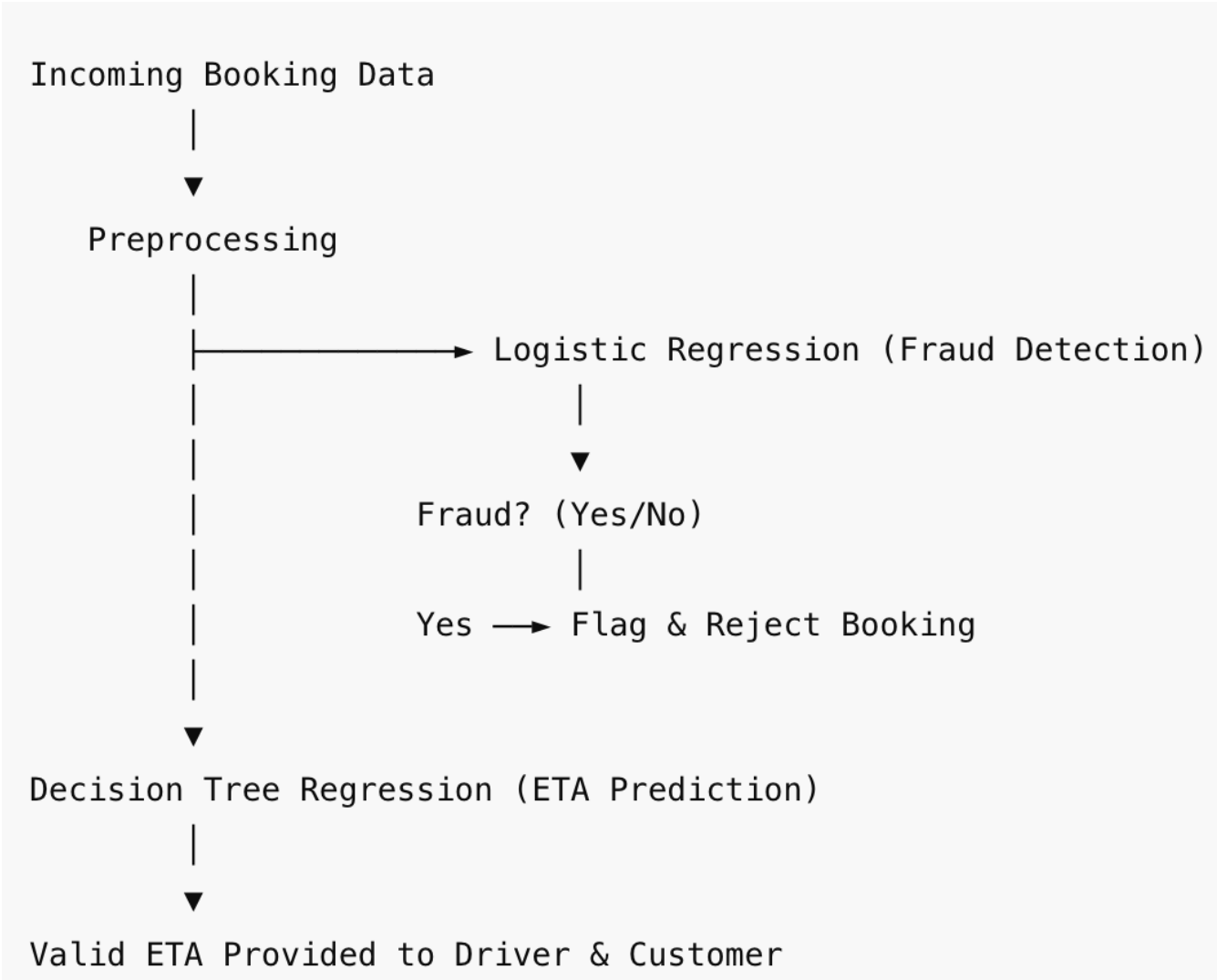
Step	Action	Goal
1	Data Cleaning & Preprocessing	Remove noise, handle missing data
2	Feature Engineering	Improve data representation to reduce underfitting
3	Data Augmentation	Increase data variety to reduce overfitting
4	Model Selection	Simpler model for overfitting, complex for underfitting
5	Regularization & Pruning	Prevent overfitting by controlling complexity
6	Hyperparameter Tuning	Optimize model performance
7	Cross-Validation	Ensure robustness and generalization
8	Evaluation	Check metrics on training and test data
9	Ensemble Methods (if needed)	Combine models to balance bias and variance
10	Deployment	Use final model on unseen data

Example Implementation

Scenario: Retail company predicting product purchase likelihood

1. Preprocess data: Handle missing prices, encode product categories, normalize sales features.
2. Feature engineering: Include seasonal trends, user behavior scores.
3. Algorithm: Random Forest (for medium complexity)
4. Hyperparameter tuning: Grid Search on number of trees and max depth.
5. Regularization: Limit tree depth to prevent overfitting.
6. Cross-validation: 5-Fold to check performance stability.
7. Evaluation: ROC-AUC, Precision, Recall
8. Optional ensemble: Combine with Gradient Boosting for better predictions.

Expected Result:

	• Balanced accuracy on training and test sets • Reduced variance and bias • Improved predictions on unseen retail data
2.	A ride-hailing company wants to (A) predict ETA (minutes) (continuous) and (B) detect fraudulent bookings (yes/no) (categorical). Design a supervised learning solution by selecting one basic regression algorithm (Linear Regression / k-NN Regression / Decision Tree Regression) for ETA and one basic classification algorithm (Logistic Regression / Decision Tree / Naive Bayes) for fraud. Create a workflow that includes preprocessing steps, model training, and evaluation. Propose how both models could be integrated into a single decision-making pipeline for the company. 

	A ride-hailing company has two tasks: predicting **ETA (continuous)** and detecting **fraudulent bookings (binary)**. For ETA prediction, a Decision Tree Regressor is chosen because it can capture non-linear relationships between features like distance, time of day, traffic, and driver experience, and does not require feature scaling. The preprocessing includes handling missing values, encoding categorical features such as day of the week or weather, and feature engineering (e.g., calculating distance or peak-hour indicators). The model is trained on 80% of the data, and performance is evaluated using MAE, RMSE, and R^2 score. For fraud detection, Logistic Regression is selected for its simplicity and interpretability, as well as its ability to provide probability estimates. Features include booking frequency, payment method, device ID, time of booking, and location patterns. Preprocessing involves imputing missing values, encoding categorical variables, scaling numeric features, and addressing class imbalance using SMOTE or class weights. Evaluation metrics include accuracy, precision, recall, F1-score, and ROC-AUC. Both models can be integrated into a single pipeline: when a booking arrives, preprocessing is applied to prepare features. First, the Logistic Regression model predicts fraud probability; bookings flagged as fraudulent are rejected. Non-fraudulent bookings are then sent to the Decision Tree Regressor to predict ETA, which is provided to the driver and customer app. This approach ensures fraudulent bookings do not waste resources while providing accurate ETA predictions for valid rides. Overfitting is controlled with feature selection, pruning, or regularization, while underfitting is reduced by including relevant features and proper model choice. Hyperparameter tuning and cross-validation further ensure robust performance on unseen data.

UNIT – IV

UNSUPERVISED LEARNING

Q. No	QUESTIONS
	PART A
1.	What are the benefits of using unsupervised learning in machine learning?

<table>
<tr><td></td><td>Unsupervised learning helps discover hidden patterns and structures in unlabeled data, such as grouping similar items (clustering) or reducing dimensionality (feature extraction).</td></tr>
<tr><td>2.</td><td>What does "manifold learning" mean?
Manifold learning is a type of unsupervised learning technique used to reduce the dimensionality of high-dimensional data while preserving its underlying structure. It assumes that the data lies on a lower-dimensional “manifold” embedded in a higher-dimensional space, and it tries to uncover that manifold to simplify analysis.</td></tr>
<tr><td>3.</td><td>Explain the difference between normalization and standardization in data preprocessing.

<table><tr><th>Aspect</th><th>Normalization</th><th>Standardization</th></tr><tr><td>Purpose</td><td>Scale data to a fixed range (0–1 or -1–1)</td><td>Center data around mean 0 with standard deviation 1</td></tr><tr><td>Formula</td><td>$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$</td><td>$x_{\text{std}} = \frac{x - \mu}{\sigma}$</td></tr><tr><td>Range</td><td>Fixed (e.g., 0–1)</td><td>Unbounded (can be negative or >1)</td></tr><tr><td>Sensitive to Outliers</td><td>Yes</td><td>Less sensitive</td></tr><tr><td>Common Use</td><td>Algorithms sensitive to scale (KNN, Neural Networks)</td><td>Algorithms assuming normally distributed data (SVM, Logistic Regression, PCA)</td></tr></table></td></tr>
<tr><td>4.</td><td>Describe the advantages of using Hierarchical Clustering.
No need to specify the number of clusters in advance
Provides a dendrogram for easy visualization of hierarchical relationships
Can capture clusters of different sizes and non-spherical shapes
Deterministic results (same output every time)
Suitable for small to medium-sized datasets\</td></tr>
<tr><td>5.</td><td>Write the purpose and applications of feature extraction.
Reduce the dimensionality of data while retaining important information
Transform raw data into meaningful features that improve model performance
Remove irrelevant or redundant information to simplify learning

Applications of Feature Extraction:

● Image Processing</td></tr>
</table>

	• Natural Language Processing (NLP) • Speech Recognition • Healthcare • Finance
6.	Suppose you have a dataset with high dimensionality. Which dimensionality reduction technique would you use and why? For a high-dimensional dataset, I would use Principal Component Analysis (PCA) as a dimensionality reduction technique. PCA transforms the original features into a smaller set of uncorrelated principal components that capture the maximum variance in the data.
7.	Describe the purpose of the Silhouette Score in clustering evaluation. The Silhouette Score measures how well each data point fits within its assigned cluster compared to other clusters. It quantifies cluster cohesion (how close points are within a cluster) and cluster separation (how far points are from other clusters). Values range from -1 to 1: • Close to 1 → well-clustered • Around 0 → on the boundary • Negative → possibly misclassified
8.	List some common challenges in unsupervised learning. Determining the number of clusters or groups in advance High dimensionality of data leading to the curse of dimensionality No labeled data for validation or performance evaluation Sensitivity to noise and outliers affecting cluster quality Scalability issues with large datasets
9.	State the advantages of hierarchical clustering over K-means.

	Aspect	**Hierarchical Clustering**	**K-Means Clustering**
	Number of Clusters	Not required in advance	Must be specified beforehand
	Output	Dendrogram showing hierarchical relationships	Flat clusters only
	Cluster Shapes	Can capture clusters of different shapes and sizes	Assumes spherical clusters
	Determinism	Deterministic (same result each run)	Non-deterministic (depends on initialization)
	Dataset Size	Suitable for small to medium datasets	Efficient for large datasets

10. State the advantage of using PCA for dimensionality reduction.

PCA reduces the number of features while retaining most of the data's variance, improving computational efficiency and simplifying analysis.

11. Differentiate between K-Means and DBSCAN.

Aspect	**K-Means**	**DBSCAN**
Cluster Definition	Partitions data into k clusters based on distance to centroids	Forms clusters based on density of points
Number of Clusters	Must be specified beforehand (k)	Automatically detects number of clusters
Cluster Shape	Assumes spherical clusters	Can detect clusters of arbitrary shape
Handling Noise	Sensitive to outliers	Can identify and exclude noise points
Determinism	Depends on initialization	Deterministic with same parameters

12. How does the choice of algorithm affect the outcome of unsupervised learning tasks?

The choice of algorithm directly affects how clusters or patterns are identified in unsupervised learning. Different algorithms have different assumptions, strengths, and limitations:

- K-Means assumes spherical, equally sized clusters; it may fail on irregularly shaped clusters or varying densities.
- DBSCAN identifies arbitrary-shaped clusters and can detect noise but may struggle with varying density regions.
- Hierarchical Clustering produces a tree of clusters and captures nested structures but can be computationally expensive for large datasets.

13.	A company wants to segment customers based on their buying behaviour. Which clustering algorithm would you recommend and why? I would recommend K-Means clustering for customer segmentation based on buying behavior. Reason: ● Customer data (e.g., purchase frequency, amount spent, product categories) is usually numerical and can be reasonably assumed to form spherical or compact clusters. ● K-Means is efficient for large datasets, making it suitable for companies with many customers.
14.	How is the Davies-Bouldin Index (DBI) used to evaluate the clustering results? The Davies-Bouldin Index (DBI) evaluates clustering quality by measuring the average similarity between each cluster and its most similar cluster. ● It considers both intra-cluster cohesion (how close points are within a cluster) and inter-cluster separation (how far clusters are from each other).
15.	A small dataset has 4 points clustered as follows: Cluster A: P1, P2 Cluster B: P3, P4 Pairwise distances between the points: **P1 P2 P3 P4** P1 0 2 7 8 P2 2 0 6 9 P3 7 6 0 3 P4 8 9 3 0 Compute the Silhouette coefficient for point P1.

	The **Silhouette coefficient** for a point is given by: $$s = \frac{b - a}{\max(a, b)}$$ Where: • a = average distance from the point to all other points in the **same cluster** • b = average distance from the point to all points in the **nearest other cluster** **Step 1: Compute a (intra-cluster distance)** • Cluster A = {P1, P2} • Distance between P1 and P2 = 2 $$a = \text{average distance to other points in same cluster} = 2$$ **Step 2: Compute b (nearest cluster distance)** • Other cluster (Cluster B) = {P3, P4} • Distances: P1–P3 = 7, P1–P4 = 8 $$b = \text{average distance to points in nearest cluster} = \frac{7 + 8}{2} = 7.5$$ **Step 3: Compute Silhouette coefficient** $$s = \frac{b - a}{\max(a, b)} = \frac{7.5 - 2}{\max(2, 7.5)} = \frac{5.5}{7.5} \approx 0.733$$
16.	Consider points along a line: 2, 3, 10, 11. Apply K-means with k=2 clusters: Cluster 1: {2, 3} Cluster 2: {10, 11} Calculate the SSE for these clusters.

Step 1: Compute the centroid of each cluster

- Centroid of Cluster 1:

$$\mu_1 = \frac{2+3}{2} = \frac{5}{2} = 2.5$$

- Centroid of Cluster 2:

$$\mu_2 = \frac{10+11}{2} = \frac{21}{2} = 10.5$$

Step 2: Compute squared distances from each point to its centroid

Cluster 1:

- $(2-2.5)^2 = (-0.5)^2 = 0.25$
- $(3-2.5)^2 = (0.5)^2 = 0.25$

SSE for Cluster 1:

$$0.25 + 0.25 = 0.5$$

Cluster 2:

- $(10-10.5)^2 = (-0.5)^2 = 0.25$
- $(11-10.5)^2 = (0.5)^2 = 0.25$

SSE for Cluster 2:

$$0.25 + 0.25 = 0.5$$

Step 3: Compute total SSE

$$\text{SSE}_{\text{total}} = 0.5 + 0.5 = 1.0$$

✅ **Answer: Total SSE = 1.0**

Q.No	QUESTIONS
	PART – B
1.	A social media company wants to group users based on how they interact with content (likes, shares, time spent, comments, etc.) to target ads better. Analyze which features should be extracted and how? Discover the type of clustering algorithms that are suitable for such high-dimensional user behavior data and explain why? **Feature Extraction for User Behavior** To group users based on social media interactions, it is essential to extract features that capture user engagement patterns. Relevant features include the number of likes, shares, and

	comments, the average time spent on posts or videos, frequency of sessions, and types of content engaged with. Additionally, derived features such as sentiment of comments, click-through rates on advertisements, and engagement with specific content categories provide deeper insights into user preferences. These features are extracted by aggregating raw interaction data over a period, normalizing numerical values to a common scale, and encoding categorical features using one-hot encoding or embeddings to make them suitable for clustering. **Clustering Algorithms for High-Dimensional Data** Social media user data is typically high-dimensional, sparse, and may contain correlated features. K-Means is suitable after applying dimensionality reduction techniques such as Principal Component Analysis (PCA) or t-SNE, which reduce the number of features and improve cluster quality. Hierarchical clustering is advantageous when nested relationships exist among users, as it does not require the number of clusters to be predefined. Density-based algorithms such as DBSCAN or HDBSCAN are effective for discovering clusters of arbitrary shapes and can handle noise or outlier users efficiently. Gaussian Mixture Models (GMMs) can probabilistically assign users to overlapping clusters, capturing the variability in user behavior. **Justification of Algorithm Choice** The selected algorithms are appropriate due to the characteristics of social media data. Dimensionality reduction mitigates issues related to high-dimensionality and correlated features, enhancing clustering performance. Hierarchical and density-based algorithms capture complex and irregular cluster shapes that may arise in user behavior patterns. Density-based approaches are particularly robust against noisy or inconsistent users. By combining well-extracted features with suitable clustering methods, social media companies can effectively segment users for targeted advertising and personalized recommendations.
2.	Analyze the role of Principal Component Analysis (PCA) in dimensionality reduction. How does PCA help in improving the performance of clustering algorithms, and what are its limitations? Principal Component Analysis (PCA) is a statistical technique used to reduce the number of features in a dataset while retaining the most important information. It transforms the original correlated features into a new set of uncorrelated variables called **principal components**, ordered by the amount of variance they explain. By selecting the top few principal components,

PCA reduces dimensionality, simplifies the dataset, and removes redundant or less informative features.

How PCA Improves Clustering Performance

PCA enhances the performance of clustering algorithms in several ways:

1. **Noise Reduction:** By removing low-variance components, PCA filters out noise, allowing clustering algorithms to focus on meaningful patterns.

2. **Computational Efficiency:** Lowering the number of dimensions reduces the computational cost for distance calculations in algorithms like K-Means.

3. **Improved Cluster Separation:** Transforming data into principal components can reveal the intrinsic structure of the dataset, making clusters more distinct and easier to identify.

4. **Mitigates the Curse of Dimensionality:** High-dimensional data can dilute distance measures, reducing clustering accuracy. PCA reduces dimensionality while preserving variance, improving clustering reliability.

Limitations of PCA

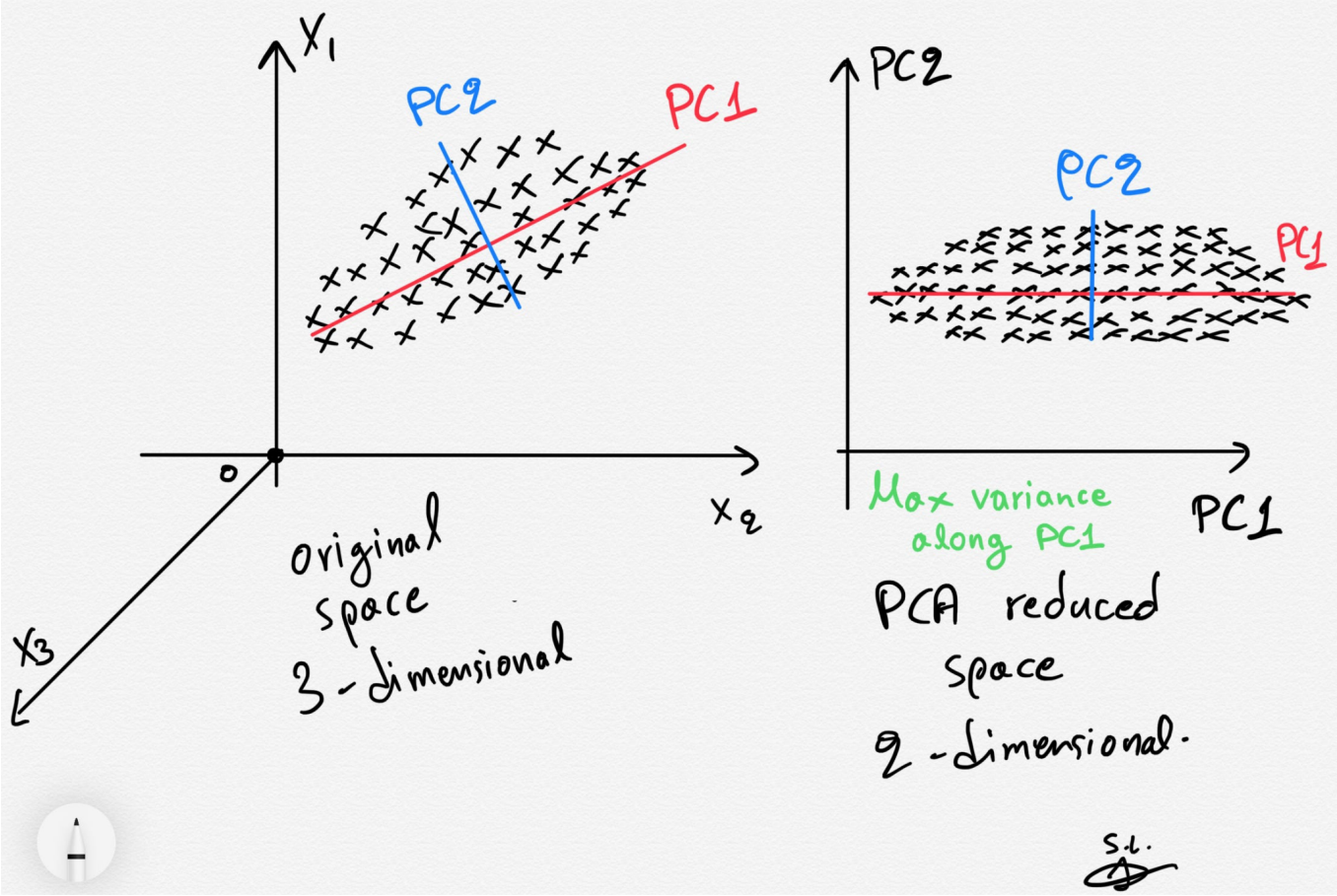
Despite its benefits, PCA has some limitations:

1. **Linear Assumption:** PCA captures only linear relationships between features and may fail to represent complex, nonlinear patterns.

2. **Loss of Interpretability:** Principal components are combinations of original features, which can make it difficult to interpret the transformed data.

3. **Variance Does Not Equal Importance:** PCA assumes that components with higher variance are more important, which may not always align with the underlying task or clustering objective.

4. **Sensitivity to Scaling:** PCA requires careful normalization; features with larger scales

	can dominate the principal components. 
3.	Evaluate manifold learning techniques for dimensionality reduction and feature extraction in complex datasets. Manifold learning is a class of **nonlinear dimensionality reduction techniques** that aim to uncover the low-dimensional structure embedded within high-dimensional data. Unlike linear methods such as PCA, manifold learning assumes that data points lie on a curved, low-dimensional manifold within the higher-dimensional space. These techniques are particularly useful for complex datasets where relationships between features are nonlinear. **Key Manifold Learning Techniques** 1. **Isomap:** ○ Extends classical Multidimensional Scaling (MDS) by preserving **geodesic distances** along the manifold.

- Captures the intrinsic geometry of the data, making it suitable for datasets with nonlinear structures.

2. **Locally Linear Embedding (LLE):**
   - Preserves **local neighborhood relationships** rather than global distances.
   - Each data point is reconstructed as a linear combination of its nearest neighbors, and low-dimensional coordinates are derived from these relationships.

3. **t-SNE (t-distributed Stochastic Neighbor Embedding):**
   - Focuses on preserving **pairwise similarities** in a probabilistic sense.
   - Widely used for visualization of high-dimensional data in 2D or 3D while maintaining local clusters.

4. **UMAP (Uniform Manifold Approximation and Projection):**
   - Preserves both **local and global structure** efficiently.
   - Scales better than t-SNE and is effective for feature extraction and clustering tasks.

Role in Dimensionality Reduction and Feature Extraction

- Manifold learning reduces dimensionality while preserving the **intrinsic nonlinear relationships** between features.
- It enables better visualization and interpretation of complex data.
- Extracted features can improve the performance of downstream tasks such as clustering, classification, and anomaly detection by revealing the underlying data structure.
- For clustering, manifold learning can transform data into a space where clusters are more separable, enhancing algorithm accuracy.

Advantages

- Captures nonlinear relationships that linear methods like PCA cannot.
- Helps in visualizing complex datasets in low-dimensional space.
- Improves performance of clustering and classification by revealing the true structure of data.

	Limitations ● Computationally intensive for large datasets (especially t-SNE and LLE). ● Sensitive to hyperparameters such as neighborhood size and perplexity. ● May distort global structure (t-SNE focuses on local structure). ● Learned features can be **hard to interpret** in terms of original variables.
4.	Evaluate the effectiveness of different clustering evaluation metrics in accessing clustering quality. Clustering is an unsupervised learning task, and evaluating the quality of clusters is crucial because there are no predefined labels. Clustering evaluation metrics help assess how well the algorithm has grouped similar data points together and separated dissimilar ones. These metrics are broadly classified into internal, external, and relative evaluation methods. Internal Evaluation Metrics Internal metrics measure clustering quality using intrinsic information from the dataset, without requiring ground truth labels. Examples include: 1. Silhouette Score: ○ Measures how similar a point is to its own cluster compared to other clusters. ○ Ranges from -1 to 1; higher values indicate better-defined clusters. ○ Effectiveness: Captures both cohesion (within-cluster similarity) and separation (between-cluster dissimilarity). Works well for spherical clusters but may be less effective for irregular shapes. 2. Davies-Bouldin Index (DBI): ○ Computes the ratio of within-cluster scatter to between-cluster separation. ○ Lower values indicate better clustering. ○ Effectiveness: Useful for comparing clusters of varying sizes but can be sensitive to noise and outliers. 3. Calinski-Harabasz Index: ○ Ratio of between-cluster dispersion to within-cluster dispersion. ○ Higher values indicate compact and well-separated clusters.

- Effectiveness: Works well for balanced clusters but may not capture complex cluster shapes.

External Evaluation Metrics

External metrics compare clustering results with ground truth labels. Examples include:

1. Adjusted Rand Index (ARI):
    - Measures agreement between predicted clusters and true labels, adjusted for chance.
    - Effectiveness: Accurate for assessing clustering when labels are known; robust to permutations of cluster labels.
2. Normalized Mutual Information (NMI):
    - Measures the shared information between cluster assignments and true labels.
    - Effectiveness: Handles varying cluster sizes; sensitive to overlapping clusters.

Relative Evaluation

Relative metrics compare clustering results across different parameter settings or algorithms, such as varying k in K-Means. They are useful for model selection, helping identify the optimal number of clusters using internal metrics like Silhouette or DBI.

5. Evaluate the performance of different clustering algorithms (e.g., K-means, DBSCAN, Hierarchical Clustering) on a dataset with varying densities and noise levels.

Clustering algorithms group similar data points without using labeled data. The performance of different algorithms varies depending on dataset characteristics, such as cluster density, shape, size, and the presence of noise or outliers. Evaluating algorithms like K-Means, DBSCAN, and Hierarchical Clustering under varying conditions helps determine their suitability for a particular dataset.

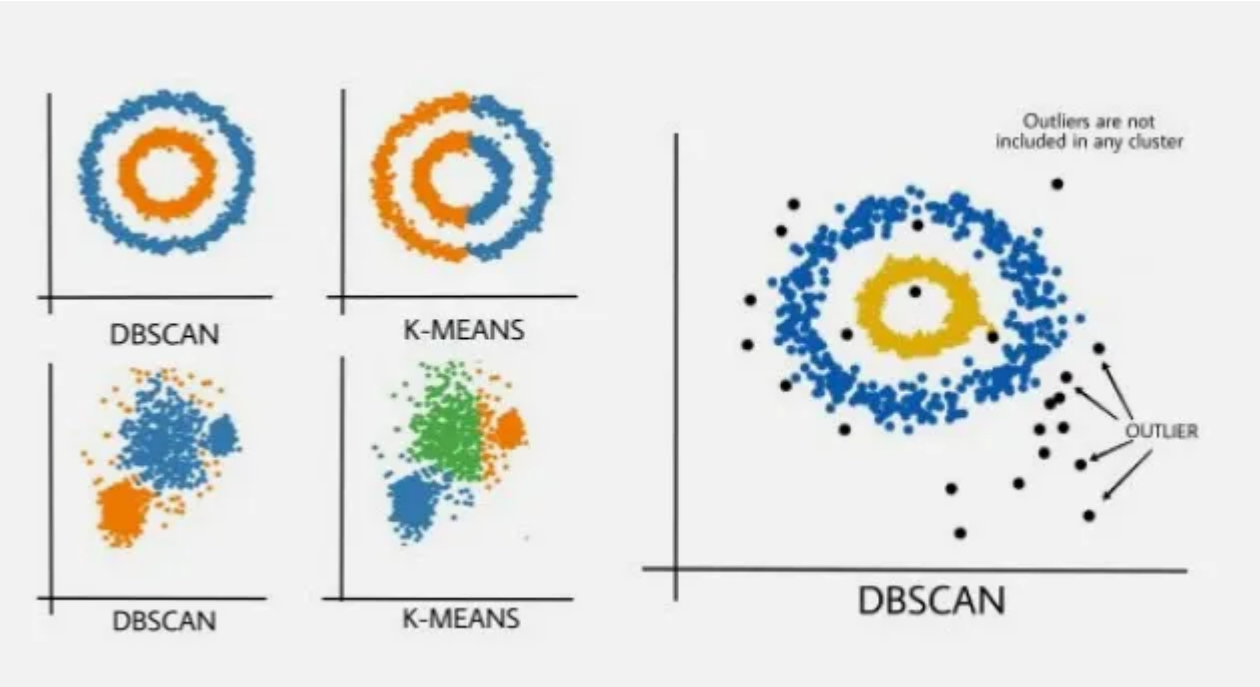
K-Means Clustering

K-Means partitions data into k clusters by minimizing the sum of squared distances from cluster centroids. It performs well on datasets with **well-separated, spherical clusters of similar size and density**. However, it struggles with:

- **Varying densities:** Sparse clusters can be merged with dense clusters, reducing accuracy.

- **Noise and outliers:** K-Means is sensitive to outliers since centroids are influenced by extreme points.

- **Irregular shapes:** Non-spherical clusters are poorly captured.
  Overall: K-Means is efficient and scalable but less effective for noisy, irregular, or unevenly dense datasets.



DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN groups points based on **density**, defining clusters as regions with a minimum number of points within a radius ε. It excels in:

- **Varying densities:** Can detect dense clusters and separate sparse areas if parameters are chosen carefully.

- **Noise handling:** Naturally identifies noise points as outliers, improving cluster quality.

- **Irregular shapes:** Captures arbitrarily shaped clusters.

Limitations include sensitivity to ε and MinPts parameters and difficulty handling clusters with **vastly differing densities** simultaneously.

Overall: DBSCAN is robust for noisy, irregular datasets but requires careful parameter tuning.

Hierarchical Clustering

Hierarchical clustering builds a tree of clusters either **agglomeratively** (bottom-up) or **divisively** (top-down). It has the advantage of:

- **No need to predefine k:** The dendrogram allows flexibility in choosing the number of clusters.
- **Capturing nested clusters:** Useful for datasets with hierarchical structure.

Limitations include:

- **Sensitivity to noise and outliers:** Single-linkage may produce chaining effects.
- **Scalability:** Computationally expensive for large datasets.
- **Varying densities:** May merge dense and sparse clusters incorrectly depending on linkage criteria.

Comparison and Evaluation

- **K-Means** is best for uniform, spherical clusters with low noise.
- **DBSCAN** is ideal for irregular clusters and noisy data but struggles with clusters of different densities.
- **Hierarchical Clustering** provides a detailed cluster hierarchy but may not handle noise or large datasets efficiently.

6.	Discuss the challenges of handling high-dimensional data in unsupervised learning. How can dimensionality reduction techniques help in addressing this challenge? High-dimensional data, such as text, images, or user behavior logs, presents several challenges for unsupervised learning algorithms like clustering or anomaly detection. One major issue is the **curse of dimensionality**, where the volume of the feature space increases exponentially with the number of dimensions. This leads to data sparsity, making distance or similarity measures less meaningful and reducing the effectiveness of clustering algorithms like K-Means. High-dimensional data also often contains **irrelevant, redundant, or noisy features**, which can obscure true patterns and degrade algorithm performance. Additionally, high dimensionality increases **computational complexity** and memory requirements, making algorithms slower and less scalable. Visualizing high-dimensional data for interpretation or validation becomes nearly impossible without some form of transformation. **Role of Dimensionality Reduction Techniques** Dimensionality reduction techniques address these challenges by transforming the data into a lower-dimensional space while preserving its essential structure. **Linear methods** like Principal Component Analysis (PCA) project data onto orthogonal axes of maximum variance, reducing redundancy and noise. **Nonlinear or manifold learning methods** such as t-SNE, Isomap, or UMAP capture complex intrinsic structures in the data. By reducing the number of dimensions: 1. Distance and similarity measures become more meaningful, improving clustering and other unsupervised learning tasks. 2. Computational efficiency and memory usage are improved, enabling faster and scalable algorithms. 3. Noise and irrelevant features are minimized, which enhances the ability to detect genuine patterns. 4. Visualization in 2D or 3D becomes possible, aiding interpretation and analysis.
7.	Evaluate the importance of data preprocessing and scaling in unsupervised learning. How can

normalization and standardization techniques impact the performance of unsupervised learning models?

Data preprocessing is a critical step in unsupervised learning because the quality of input data directly affects the ability of algorithms like clustering and dimensionality reduction to detect meaningful patterns. Raw data often contains **missing values, noise, outliers, or inconsistent formats**, which can mislead algorithms and produce inaccurate clusters or embeddings. Preprocessing ensures that data is clean, consistent, and suitable for analysis, improving the reliability and interpretability of results.

Role of Scaling in Unsupervised Learning

Scaling is particularly important in unsupervised learning because many algorithms, such as K-Means or hierarchical clustering, rely on **distance metrics** (e.g., Euclidean distance) to group similar data points. Features with larger numeric ranges can dominate the distance calculation, skewing clustering results, while features with smaller ranges may be ignored. Scaling ensures that all features contribute appropriately to similarity measures.

Normalization and Standardization Techniques

1. **Normalization (Min-Max Scaling):**
    - Transforms features to a common range, typically [0,1].
    - Useful when features have different units or ranges.
    - Helps algorithms like K-Means produce balanced clusters.
2. **Standardization (Z-score Scaling):**
    - Centers features by subtracting the mean and scales by the standard deviation.
    - Produces features with zero mean and unit variance.
    - Suitable for datasets with outliers and varying distributions, improving the

	performance of algorithms sensitive to variance. **Impact on Unsupervised Learning Performance** ● Improves **cluster quality** by preventing dominance of high-range features. ● Enhances **convergence speed** for iterative algorithms like K-Means. ● Reduces **bias introduced by scale differences**, enabling more accurate distance calculations. ● Enables meaningful **dimensionality reduction**, as variance-based techniques like PCA perform better on standardized data.
8.	Analyze the preprocessing steps required to make a retail customer dataset (10,000 records with purchase frequency, total spending, and product categories) suitable for clustering. Analyze and justify the choice of an appropriate clustering algorithm for this task. **Preprocessing Steps for Retail Customer Dataset** To prepare a retail customer dataset with 10,000 records containing purchase frequency, total spending, and product categories for clustering, several preprocessing steps are necessary: 1. **Handling Missing Values:** ○ Check for missing entries in any feature and fill them using appropriate methods. For numeric features like purchase frequency or total spending, missing values can be imputed with the mean or median. For categorical features such as product categories, the most frequent category or a placeholder can be used. 2. **Encoding Categorical Features:** ○ Product categories are categorical and must be converted into numerical format

for clustering. One-hot encoding is commonly used to represent each category as a binary vector, ensuring that distances between points are meaningful.

3. **Scaling Numeric Features:**

    - Features like purchase frequency and total spending may have different ranges. Normalization or standardization ensures that one feature does not dominate distance calculations in algorithms like K-Means. Standardization (Z-score scaling) is particularly effective if there are outliers in spending data.

4. **Outlier Detection and Treatment:**

    - High-spending customers or extremely frequent shoppers may distort cluster centroids. Outliers can be capped, transformed (e.g., log transformation), or treated separately.

5. **Dimensionality Reduction (Optional):**

    - If one-hot encoding introduces many dimensions for product categories, dimensionality reduction techniques like PCA or t-SNE can be applied to reduce feature space while preserving variance, improving clustering efficiency.

Choice of Clustering Algorithm

For this dataset, the following considerations guide the choice of algorithm:

1. **K-Means:**

    - Efficient for large datasets and works well if clusters are roughly spherical and similar in size.

    - Sensitive to outliers and requires specifying the number of clusters in advance.

2. **DBSCAN:**

	○ Can identify clusters of arbitrary shape and handle noise, but may struggle if clusters have varying densities. ○ Parameter tuning (ε and MinPts) can be challenging for high-dimensional one-hot encoded data. 3. **Hierarchical Clustering:** ○ Useful for understanding nested relationships between customer segments but computationally expensive for 10,000 records. K-Means is the most appropriate for this retail dataset because: ● The dataset is large (10,000 records), making K-Means computationally efficient. ● Features after preprocessing (scaled numeric features and possibly reduced-dimensional encoded categories) are suitable for Euclidean distance-based clustering. ● It allows easy segmentation of customers into predefined groups for marketing strategies (e.g., high-spenders, frequent buyers, or product-preference groups).

Q. No	QUESTIONS
	PART – C
1.	A university collects data on student attendance, participation in activities, internal marks, and assignment submissions. There are no labels such as “high/low performers”. Analyze which type of unsupervised learning would be suitable and why? Discover the preprocessing challenges that can arise from academic records. Discover how the university can use clustering to design academic interventions? Since the university dataset contains student attendance, participation, internal marks, and assignment submissions without any predefined labels, unsupervised learning techniques are appropriate. Specifically, clustering is the most suitable method because it groups students into

segments based on similarity in academic behavior and engagement patterns. Clustering can reveal hidden patterns such as high performers, at-risk students, or students who participate actively but score lower academically, enabling targeted interventions.

Preprocessing Challenges

Academic records can pose several preprocessing challenges:

1. **Heterogeneous Data Types:**
    - Attendance and assignment submissions may be numeric percentages or counts, while participation could be categorical (e.g., club membership, workshop attendance). Combining these features requires encoding categorical data numerically.
2. **Missing or Incomplete Data:**
    - Students may have missing internal marks or incomplete assignment submissions. Imputation techniques such as mean, median, or zero-filling may be required.
3. **Different Feature Scales:**
    - Attendance might range from 0–100%, marks from 0–50, and participation scores on a 1–5 scale. Without scaling, features with larger ranges may dominate distance-based clustering algorithms. Standardization or normalization is necessary.
4. **Outliers:**
    - Students with unusually high or low attendance or marks can skew cluster centroids. Outlier detection or robust scaling may be needed.
5. **Dimensionality Issues (if many features):**

- If additional features like activity types or course-specific grades are included, the dataset may become high-dimensional, requiring dimensionality reduction (e.g., PCA) before clustering.

Using Clustering for Academic Interventions

Once clusters are identified, the university can design targeted interventions:

1. **At-Risk Students:**
    - Students with low attendance, low participation, and poor marks can be grouped and provided with mentoring, tutoring, or counseling sessions.
2. **High Performers:**
    - Clusters of high-achieving students can be engaged with advanced projects, leadership roles, or scholarships.
3. **Participation-Focused Interventions:**
    - Students with low engagement but moderate marks can be encouraged to join clubs, workshops, or collaborative activities to improve soft skills.
4. **Personalized Feedback:**
    - Clusters enable faculty to tailor feedback, resources, or intervention programs specific to each group, improving overall academic performance and engagement.

Clustering is an effective unsupervised learning technique for analyzing student academic behavior when labels are unavailable. Preprocessing steps such as handling missing values, scaling, encoding categorical data, and managing outliers are crucial for meaningful clustering. By segmenting students into clusters, the university can design targeted academic interventions, enhance student engagement, and improve performance outcomes.

2	Create a comprehensive example of an unsupervised learning problem that utilizes clustering techniques. Evaluate how you would approach solving this problem and assess the challenges that might arise during the clustering process. Consider an e-commerce company that wants to segment its customers to design targeted marketing strategies. The company collects data on each customer, including purchase frequency, average spending per order, preferred product categories, website browsing time, and engagement with promotions. There are no labels indicating high-value or low-value customers. The objective is to identify distinct customer segments to improve personalization and increase sales. This is a classic unsupervised learning problem that can be solved using clustering techniques. **Approach to Solving the Problem** 1. **Data Collection and Exploration:** ○ Aggregate raw transaction logs, website analytics, and engagement data. ○ Explore data distributions, check for missing values, and understand feature ranges. 2. **Preprocessing:** ○ **Handle missing values:** Impute or remove incomplete records. ○ **Encode categorical features:** Convert product categories into numerical representations using one-hot encoding or embeddings. ○ **Scaling:** Standardize numeric features like spending, browsing time, and purchase frequency to ensure all features contribute equally in distance calculations. ○ **Outlier treatment:** Identify extreme spenders or rare behaviors and decide whether to transform, cap, or retain them depending on business goals.

3. **Dimensionality Reduction (Optional):**

    - If the number of features is high, apply PCA or t-SNE to reduce dimensionality while preserving variance, improving clustering efficiency.

4. **Clustering Algorithm Selection:**

    - **K-Means:** Efficient for large datasets and works well if clusters are roughly spherical.

    - **DBSCAN:** Useful if customer segments have irregular shapes and some noise exists.

    - **Hierarchical Clustering:** Can identify nested relationships between segments but may be computationally intensive for very large datasets.

5. **Cluster Evaluation:**

    - Use **internal metrics** like Silhouette Score, Davies-Bouldin Index, or Calinski-Harabasz Index to determine the quality of clusters.

    - Visualize clusters (using PCA or t-SNE projections) to interpret results.

6. **Interpretation and Application:**

    - Identify high-value, medium-value, and low-value customer segments.

    - Design targeted marketing campaigns, personalized discounts, or engagement strategies for each cluster.

Challenges in the Clustering Process

1. **Feature Heterogeneity:**

- Combining numeric, categorical, and derived features can complicate distance calculations.

2. **High Dimensionality:**

   - Many features, especially from one-hot encoding, can make distance measures less meaningful and increase computational cost.

3. **Determining the Number of Clusters:**

   - Algorithms like K-Means require specifying k, which may not be obvious. Internal metrics or the elbow method can help but are not always definitive.

4. **Outliers and Noise:**

   - Extreme behavior (e.g., very high spending) can skew cluster centroids, especially for K-Means.

5. **Cluster Shape Variability:**

   - Customer segments may not be spherical; density-based methods may be more appropriate but require careful parameter tuning.

6. **Interpretability:**

   - Understanding what defines each cluster in business terms can be challenging if many features or dimensionality reduction techniques are used.

UNIT V - APPLIED MACHINE LEARNING AND EVALUATION

Q.No	QUESTIONS
	PART A

1.	Define categorical variables with examples. **Categorical variables:** Variables that take discrete values representing categories. *Example:* Color = {Red, Blue, Green}, Gender = {Male, Female}.
2.	What is one-hot encoding? **One-hot encoding:** Converts categorical variables into binary vectors where each category is represented by a separate column with 1 for presence and 0 for absence.
3.	Explain the difference between label encoding and one-hot encoding. Label encoding assigns an integer to each category (e.g., Red=0, Blue=1). One-hot encoding creates binary columns for each category (e.g., Red=[1,0,0]).
4.	Apply one-hot encoding to the variable *Color = {Red, Blue, Green}* **One-hot encoding Color = {Red, Blue, Green}:** **Color** | **Red** | **Blue** | **Green** Red | 1 | 0 | 0 Blue | 0 | 1 | 0 Green | 0 | 0 | 1
5.	List three methods of feature selection. Filter methods (e.g., correlation, chi-square) Wrapper methods (e.g., forward/backward selection) Embedded methods (e.g., Lasso, decision tree importance)
6.	Apply correlation-based feature selection on a dataset with 10 variables. Compute pairwise correlation between 10 variables and remove highly correlated features to reduce redundancy.
7.	Justify the need for feature selection in reducing overfitting. Removing irrelevant or redundant features simplifies the model, reducing complexity and preventing it from fitting noise in the training data.
8.	Propose a hybrid feature selection approach for a medical dataset. Combine filter methods (correlation or chi-square) with wrapper methods (recursive feature elimination) to select relevant features.

9.	What is cross-validation? A technique to evaluate model performance by splitting data into multiple training and validation sets, ensuring all data is used for both training and testing.
10.	Apply 5-fold cross-validation on a dataset with 1000 records. How many records are in each fold? **5-fold cross-validation on 1000 records:** Each fold has 1000 ÷ 5 = **200 records**.
11.	Analyze the advantages of cross-validation over a single train-test split. Reduces bias from a single random split. Provides more reliable and stable performance estimates.
12.	Evaluate the impact of increasing the number of folds on bias and variance. Higher folds → lower bias but higher variance. Fewer folds → higher bias but lower variance.
13.	Define hyperparameter tuning. The process of selecting the best combination of parameters (not learned during training) to optimize model performance.
14.	Perform a grid search to find the best learning rate and depth for a decision tree. Test combinations of learning_rate = [0.01, 0.1, 0.2] and max_depth = [3,5,7] to find the combination yielding highest validation accuracy.
15.	Compare grid search with random search. Grid search evaluates all combinations exhaustively. Random search evaluates a random subset, faster for large search spaces.
16.	List three classification metrics and two regression metrics. ● Classification: Accuracy, Precision, Recall ● Regression: Mean Squared Error (MSE), R^2 score

Q.No	**QUESTIONS**
	PART – B
1.	Analyze the challenges of handling high-cardinality categorical variables. Compare different encoding strategies and justify which is more effective for real-world datasets such as customer

segmentation.

High-cardinality categorical variables are those with a large number of unique categories, such as customer IDs, product SKUs, or ZIP codes. Handling such variables poses several challenges:

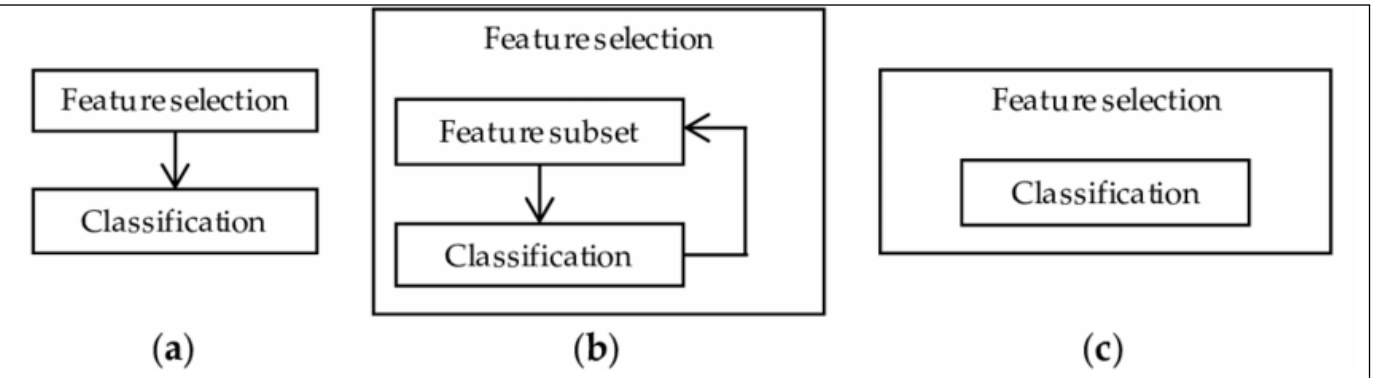
1. **Curse of Dimensionality:** Encoding methods like one-hot encoding create a new column for each unique category, drastically increasing the feature space. This leads to sparse, high-dimensional data, which can slow down training and increase memory usage.
2. **Overfitting Risk:** Models may memorize rare categories rather than generalize patterns, particularly in small datasets.
3. **Scalability Issues:** Algorithms that rely on distance metrics (e.g., K-Means) may become inefficient when many dimensions are added due to high cardinality.

4. **Interpretability:** With thousands of encoded columns, it becomes difficult to interpret feature importance or patterns.

Encoding Strategies for High-Cardinality Variables

1. **One-Hot Encoding:**
    - Creates binary columns for each category.
    - **Pros:** Simple, interpretable, widely supported.
    - **Cons:** Not feasible for high-cardinality variables due to dimensionality explosion and sparsity.
2. **Label Encoding:**
    - Assigns a unique integer to each category.
    - **Pros:** Efficient, no dimensionality increase.
    - **Cons:** Imposes ordinal relationships where none exist, which can mislead models.
3. **Target Encoding (Mean Encoding):**
    - Replaces each category with the mean of the target variable (for supervised tasks) or aggregated statistics.
    - **Pros:** Reduces dimensionality, captures category impact on target.

	○ **Cons:** Risk of data leakage if not applied carefully; less interpretable. 4. **Frequency Encoding:** ○ Replaces categories with their occurrence frequency in the dataset. ○ **Pros:** Simple, keeps dimensionality low, can reflect importance of categories. ○ **Cons:** May not capture relationships with the target variable directly. 5. **Embedding Layers (for Deep Learning):** ○ Maps categories to dense, low-dimensional vectors learned during training. ○ **Pros:** Captures semantic similarity between categories; scalable for very high cardinality. ○ **Cons:** Requires larger datasets and more complex models. For real-world datasets like **customer segmentation**, high-cardinality features such as customer ID, ZIP code, or product type are common. ● **One-hot encoding** is impractical due to dimensionality explosion. ● **Label encoding** may introduce misleading ordinal relationships. ● **Frequency or target encoding** reduces dimensionality and preserves meaningful information, making them suitable for traditional machine learning models. ● **Embeddings** are highly effective when using neural networks, as they capture latent relationships between categories and scale well to millions of unique values.
2.	Demonstrate how feature selection improves model performance. Implement and explain Filter, Wrapper, and Embedded methods with suitable datasets. Feature selection improves model performance by: 1. **Reducing Overfitting:** Removing irrelevant or noisy features prevents the model from fitting spurious patterns. 2. **Improving Accuracy:** Focusing on informative features enhances predictive power. 3. **Reducing Computation:** Fewer features decrease training time and memory usage. 4. **Enhancing Interpretability:** Simplified models are easier to understand and analyze.



(a) Filter, (b) wrapper, and (c) embedded feature selection methods.

Feature Selection Methods

Feature selection methods are broadly classified into **Filter, Wrapper, and Embedded methods**.

1. Filter Methods

- **Concept:** Select features based on statistical measures without involving the learning algorithm.
- **Example Dataset:** Breast Cancer dataset (numeric features like radius, texture, perimeter).
- **Implementation:**

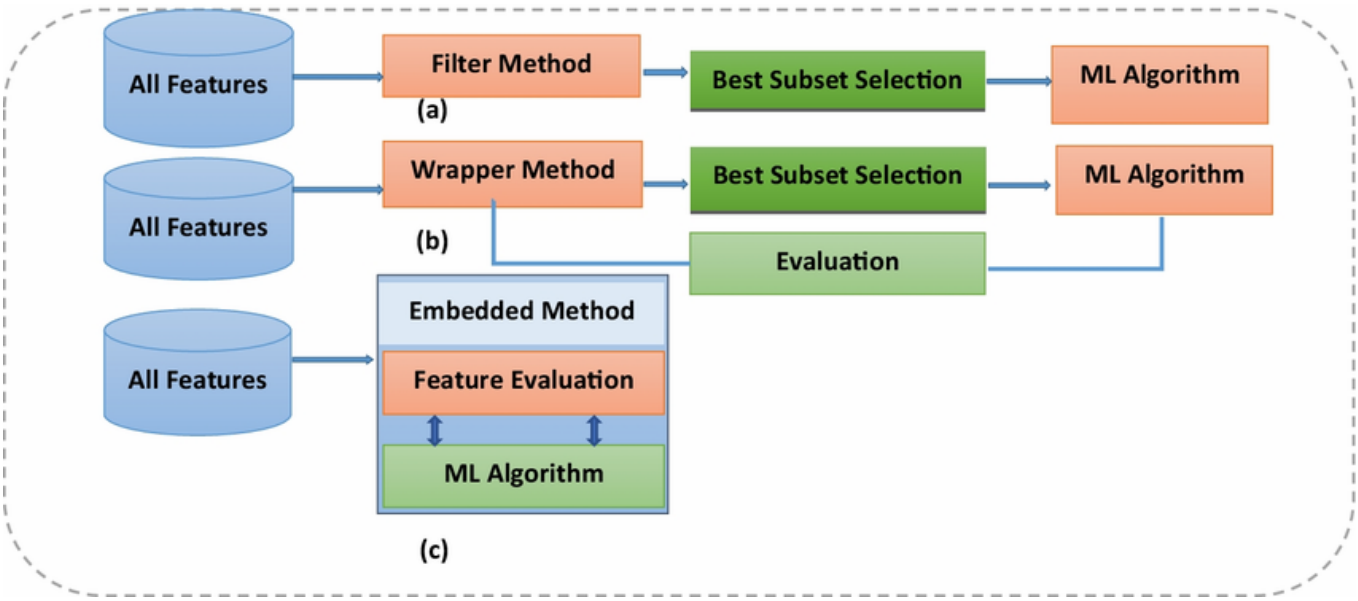
```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectKBest, f_classif

data = load_breast_cancer()
X, y = data.data, data.target

# Select top 5 features based on ANOVA F-value
selector = SelectKBest(score_func=f_classif, k=5)
X_new = selector.fit_transform(X, y)
print("Selected feature indices:", selector.get_support(indices=True))
```

2. Wrapper Methods

	• **Concept:** Evaluate subsets of features by training a model and selecting the combination that maximizes performance.
	• **Example Dataset:** Diabetes dataset.
	
	Implementation:
	
	from sklearn.datasets import load_diabetes
	from sklearn.feature_selection import RFE
	from sklearn.linear_model import LinearRegression
	data = load_diabetes()
	X, y = data.data, data.target
	model = LinearRegression()
	rfe = RFE(model, n_features_to_select=5)
	X_new = rfe.fit_transform(X, y)
	print("Selected feature indices:", rfe.get_support(indices=True))
	
	3. Embedded Methods
	
	• **Concept:** Feature selection occurs during model training. The model itself decides which features are important.
	• **Example Dataset:** Boston Housing dataset (regression).
	
	from sklearn.datasets import load_boston
	from sklearn.ensemble import RandomForestRegressor
	import numpy as np
	
	data = load_boston()
	X, y = data.data, data.target
	
	model = RandomForestRegressor()
	model.fit(X, y)
	importances = model.feature_importances_
	selected_features = np.argsort(importances)[-5:] # top 5 features
	print("Selected feature indices:", selected_features)

3.	Compare and contrast Filter, Wrapper, and Embedded methods of feature selection. Analyze

their strengths and weaknesses in high-dimensional datasets like genomics or text classification.



Aspect	Filter Methods	Wrapper Methods	Embedded Methods
Mechanism	Evaluate features using statistical measures independent of the learning algorithm (e.g., correlation, Chi-square, mutual information)	Evaluate subsets of features by training a model and selecting the subset that maximizes performance (e.g., RFE, forward/backward selection)	Feature selection occurs **during model training**, where the algorithm naturally ranks or penalizes features (e.g., Lasso, decision trees, Random Forests)
Feature Interaction	Ignores interactions; evaluates features individually	Considers feature interactions; evaluates combinations of features	Considers interactions in the context of the model
Computational Cost	Low; scalable to very high-dimensional data	High; evaluates many subsets; computationally expensive for large feature sets	Moderate; efficient as selection is integrated into training
Scalability	Excellent for large datasets (e.g., genomics, text)	Poor for high-dimensional data; may require prior filtering	Good; suitable for high-dimensional data with regularization or tree-based models
Accuracy / Performance	Moderate; may miss important feature combinations	High; usually yields optimal predictive performance	High; performance depends on the model and regularization
Overfitting Risk	Low; simple selection prevents overfitting	High; repeated model evaluation may overfit small datasets	Moderate; embedded selection reduces overfitting compared to wrapper
Interpretability	High; easy to understand feature importance	Moderate; subset selection may obscure individual feature contribution	Moderate; model-specific; feature importance depends on the algorithm
Examples	Correlation-based selection, ANOVA F-test, Chi-square test	Recursive Feature Elimination (RFE), forward/backward selection	Lasso (L1 regularization), Decision Trees, Random Forest feature importance
Strengths	Fast, simple, model-agnostic, good for initial filtering	Captures feature interactions; often more accurate	Efficient, considers interactions, integrated with training, less prone to overfitting
Weaknesses	Ignores interactions; may select irrelevant features	Computationally expensive; may overfit; not scalable	Model-specific; may not generalize across models
Best Use in High-Dimensional Data	Initial feature reduction to remove irrelevant/redundant features in genomics or text	After filtering to refine subset of important features; small to medium datasets	Large high-dimensional datasets with regularization or ensemble methods (e.g., Lasso for genes, Random Forest for text features)

4. Analyze the impact of different encoding strategies (Label, One-Hot, Target Encoding) on both linear and tree-based models. Provide case-based examples and justify your conclusions.

Categorical variables are common in real-world datasets, and their encoding directly affects model performance. Choosing an appropriate encoding strategy depends on the **type of model** (linear vs tree-based) and the characteristics of the categorical feature. Common encoding strategies include **Label Encoding, One-Hot Encoding, and Target Encoding**.

1. Label Encoding

- **Mechanism:** Assigns a unique integer to each category (e.g., Red=0, Blue=1, Green=2).

- **Impact on Models:**
    - **Linear Models (e.g., Logistic Regression, Linear Regression):** Can mislead the model by introducing an ordinal relationship where none exists, potentially biasing coefficients.
    - **Tree-Based Models (e.g., Decision Trees, Random Forests):** Less problematic, as splits are based on category thresholds, not numeric magnitude.
- **Case Example:** Predicting customer churn using Customer Type = {Basic, Premium, VIP}. Label encoding might confuse logistic regression into assuming VIP > Premium > Basic, but tree-based models handle this correctly.

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

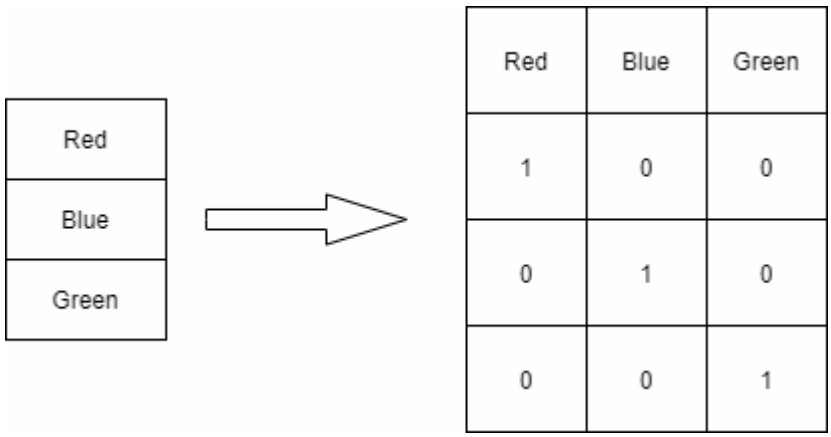
Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

2. One-Hot Encoding

- **Mechanism:** Converts categories into binary vectors, one column per category.
- **Impact on Models:**
    - **Linear Models:** Performs well because each category is independent, preventing false ordinal relationships.

- **Tree-Based Models:** Often unnecessary for trees; can increase dimensionality, which may slightly slow training but does not reduce accuracy.

- **Case Example:** Predicting loan approval based on Region = {North, South, East, West}. One-hot encoding allows linear regression to assign separate weights to each region, improving model accuracy. For a decision tree, splitting on Region directly is sufficient, so one-hot is redundant.



3. Target Encoding (Mean Encoding)

- **Mechanism:** Replaces each category with the mean value of the target variable for that category.

	Fruit	Target	Fruit (Target Encoding)
0	Apple	2	3.0
1	Apple	4	3.0
2	Apple	3	3.0
3	Orange	2	1.5
4	Orange	1	1.5

- **Impact on Models:**

	○ **Linear Models:** Captures the relationship between category and target, enhancing predictive power. ○ **Tree-Based Models:** Works well and reduces dimensionality compared to one-hot, but can introduce **data leakage** if applied incorrectly. Cross-validation or smoothing is needed to prevent overfitting. ● **Case Example:** Predicting Purchase Amount using Product Category. Target encoding replaces each category with its average purchase amount, providing a continuous numeric feature that improves linear regression predictions. For tree-based models, target encoding can accelerate training by reducing columns.
5.	Evaluate the necessity of cross-validation in imbalanced datasets (e.g., fraud detection). Which strategy is most reliable and why? In **imbalanced datasets**, such as fraud detection, the number of positive cases (fraud) is significantly smaller than negative cases (non-fraud). Evaluating model performance on such datasets using a **single train-test split** may lead to misleading results, as the rare class may be underrepresented in the test set. **Cross-validation** ensures more reliable performance estimation by repeatedly splitting the data and averaging results. **Necessity of Cross-Validation in Imbalanced Datasets** 1. **Reduces Sampling Bias:** ○ A single train-test split may omit rare classes in either the training or test set, skewing metrics like accuracy. ○ Cross-validation ensures that each observation, including rare events, is used for both training and validation across different folds. 2. **Provides Stable Performance Estimates:**

- Averaging metrics across folds reduces variance and gives a more robust measure of model generalization.

- Especially important in fraud detection where false negatives (missed frauds) are critical.

3. **Facilitates Model Tuning:**

   - Hyperparameter tuning using cross-validation avoids overfitting to a single split.

   - Helps evaluate models using metrics suitable for imbalanced data (e.g., Precision, Recall, F1-score, AUC).

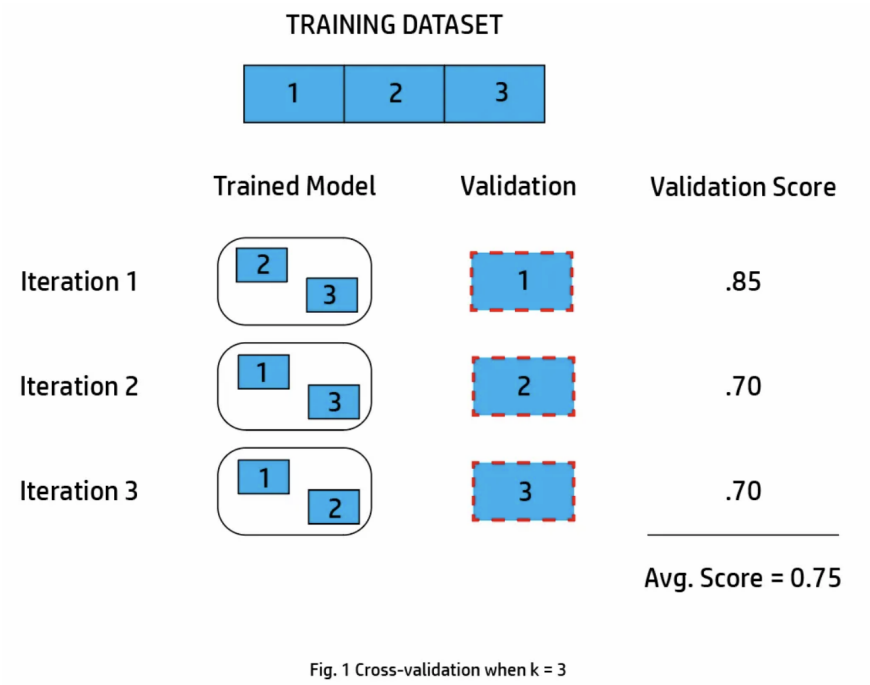


Fig. 1 Cross-validation when k = 3

Reliable Cross-Validation Strategies for Imbalanced Data

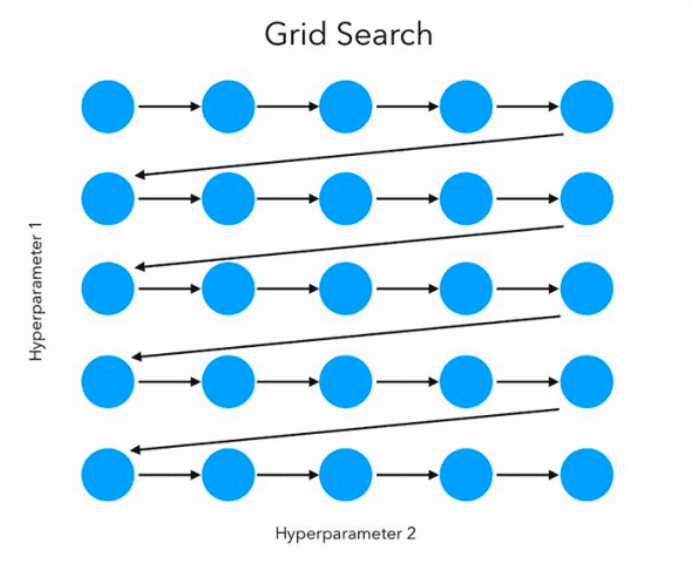
1. **Stratified K-Fold Cross-Validation:**

   - Ensures that each fold maintains the same class distribution as the original dataset.

	○ Preserves rare classes in both training and validation sets, providing more reliable estimates. 2. **Repeated Stratified K-Fold:** ○ Repeats stratified K-Fold multiple times with different random splits to further reduce variance. 3. **Time-Series or Group-Based CV (if applicable):** ○ For datasets with temporal or grouped structure, ensures that training data precedes test data chronologically or that grouped instances are not split. **Case Example: Fraud Detection** ● Dataset: 100,000 transactions, 1% fraud. ● Standard 80-20 split might result in only 200 fraud cases in the test set. A random split could yield folds with too few fraud cases. ● Using **Stratified 5-Fold CV**: each fold preserves the 1% fraud proportion (e.g., ~200 fraud cases per fold). ● Metrics such as F1-score or ROC-AUC calculated on each fold are more representative of real-world model performance.
6.	Apply Grid Search to tune hyperparameters of a Support Vector Machine classifier. Illustrate the steps with an example dataset. Support Vector Machine (SVM) classifiers have several **hyperparameters** that significantly affect performance, including: ● C (regularization parameter) ● kernel (linear, polynomial, RBF)

- gamma (for RBF/polynomial kernels, controls influence of a single training example)

Grid Search systematically searches over a predefined set of hyperparameter values to find the combination that optimizes model performance using cross-validation.



Step-by-Step Procedure

1. Import Libraries and Load Dataset

We use the **Iris dataset** as an example:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score


# Load dataset
data = load_iris()
X, y = data.data, data.target


# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Define Hyperparameter Grid

We specify ranges of hyperparameters to explore:

```
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 0.1, 1]
}
```

3. Initialize SVM and Grid Search

```
# Initialize SVM classifier
svm = SVC()
# Apply Grid Search with 5-fold cross-validation
grid_search = GridSearchCV(estimator=svm, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

4. Evaluate Best Parameters and Model Performance

```
# Best hyperparameters
print("Best Hyperparameters:", grid_search.best_params_)
# Evaluate on test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", accuracy)
```

Example Output (illustrative):

```
Best Hyperparameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Test Accuracy: 0.9667
```

7.	Compare Grid Search with Random Search and Bayesian Optimization. Analyze their effectiveness in terms of computational cost and performance in large-scale ML problems. Hyperparameter tuning is crucial in machine learning to optimize model performance. Common strategies include Grid Search, Random Search, and Bayesian Optimization. Their effectiveness depends on dataset size, number of hyperparameters, and computational resources. 1. Grid Search ● Mechanism: Exhaustively evaluates all possible combinations of hyperparameters in a predefined grid. ● Computational Cost: Very high for large grids or high-dimensional hyperparameter spaces; cost grows exponentially with the number of hyperparameters. ● Performance: Guarantees finding the optimal combination within the specified grid, but may miss better values not in the grid. ● Example: Tuning SVM with C = [0.1,1,10], kernel = ['linear','rbf'], gamma = ['scale',0.1] involves evaluating 3×2×2 = 12 models. Pros: Simple, exhaustive, and easy to implement. Cons: Not scalable for large datasets or many hyperparameters; can waste resources on unimportant regions. 2. Random Search ● Mechanism: Samples hyperparameter combinations randomly from specified distributions instead of exhaustively exploring all. ● Computational Cost: Much lower than Grid Search for large spaces; can specify the number of iterations. ● Performance: Often finds near-optimal solutions faster, especially when only a few

hyperparameters significantly influence performance.

- Example: Sampling 20 random combinations from the same SVM grid may discover better hyperparameters than a coarse grid search.

Pros: Efficient, scalable, effective in high-dimensional spaces, and avoids unnecessary evaluations.

Cons: No guarantee of finding the global optimum; results vary depending on random seed.

3. Bayesian Optimization

- Mechanism: Builds a probabilistic surrogate model (e.g., Gaussian Process) of the objective function and iteratively selects hyperparameters to maximize expected improvement.

- Computational Cost: Higher per iteration than Random Search due to model updates, but fewer evaluations are needed overall.

- Performance: Very efficient for expensive models; typically finds better hyperparameters with fewer iterations compared to Grid or Random Search.

- Example: Tuning XGBoost hyperparameters (learning_rate, max_depth, subsample) using Bayesian Optimization can quickly converge to the optimal set without testing every combination.

Analysis for Large-Scale ML Problems

1. **Grid Search:** Impractical due to exponential growth in computation; only feasible for small parameter grids.

2. **Random Search:** Efficient for large-scale problems; performs well when a few hyperparameters dominate performance.

3. **Bayesian Optimization:** Most effective when model evaluation is expensive (e.g., deep

	learning); balances exploration and exploitation to minimize total runs.

8.	Evaluate alternative performance measures to accuracy when dealing with imbalanced datasets. Illustrate with suitable examples from applications like spam detection and medical diagnosis.
	
	In **imbalanced datasets**, the number of samples in one class (minority) is significantly smaller than the other class (majority). Examples include **spam detection**, where spam emails are fewer than non-spam emails, and **medical diagnosis**, where disease cases are rare. In such cases, **accuracy is misleading**, as a model predicting only the majority class can achieve high accuracy while failing to detect the minority class. Therefore, alternative performance measures are necessary.
	
	1. Precision
	• **Definition:** Fraction of correctly predicted positive instances among all predicted positives.
	$$\text{Precision} = \frac{TP}{TP + FP}$$
	• **Use Case:** Important when **false positives are costly**.
	  • **Example:** Spam detection – high precision ensures that emails marked as spam are indeed spam, avoiding misclassification of legitimate emails.
	
	2. Recall (Sensitivity or True Positive Rate)
	• **Definition:** Fraction of correctly predicted positive instances among all actual positives.
	$$\text{Recall} = \frac{TP}{TP + FN}$$
	• **Use Case:** Important when **false negatives are costly**.
	  • **Example:** Medical diagnosis – high recall ensures that most disease cases are detected, reducing the risk of missing patients with the condition.
	
	3. F1-Score
	• **Definition:** Harmonic mean of precision and recall.
	$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
	• **Use Case:** Balances precision and recall, useful when both false positives and false negatives matter.
	  • **Example:** Detecting fraudulent transactions – both incorrectly flagged transactions (FP) and missed frauds (FN) are undesirable.

Accuracy	Predictions/ Classifications	$\frac{\text{Correct}}{\text{Correct + Incorrect}}$
Precision	Predictions/ Classifications	$\frac{\text{True Positive}}{\text{True Positive + False Positive}}$
Recall	Predictions/ Classifications	$\frac{\text{True Positive}}{\text{True Positive + False Negative}}$
F1	Predictions/ Classifications	$\frac{\text{2 * True Positive}}{\text{True Positive + 0.5 (False Positive + False Negative)}}$
IoU	Object Detections/ Segmentations	$\frac{\text{Pixel Overlap}}{\text{Pixel Union}}$ 

4. Area Under ROC Curve (AUC-ROC)

- **Definition:** Measures the ability of a classifier to distinguish between classes across different thresholds.
- **Use Case:** Evaluates model discrimination regardless of class imbalance.
  - **Example:** Predicting rare disease cases – AUC reflects how well the model ranks patients by risk.

---

5. Matthews Correlation Coefficient (MCC)

- **Definition:** Correlation between observed and predicted binary classifications; considers all four confusion matrix components.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **Use Case:** Robust metric for highly imbalanced datasets.
  - **Example:** Rare cancer detection – MCC gives a single score reflecting overall prediction quality.

Q. No	QUESTIONS
	PART – C
1.	Design a cross-validation strategy for a highly imbalanced dataset (e.g., fraud detection). Justify your choice and explain how it improves model reliability compared to simple train-test split. In highly imbalanced datasets, such as fraud detection, the minority class (fraudulent

transactions) is much smaller than the majority class (legitimate transactions). Using a simple train-test split may result in the minority class being underrepresented in either set, producing misleading performance metrics. A cross-validation strategy tailored to imbalance ensures robust and reliable model evaluation.

Proposed Cross-Validation Strategy

1. Stratified K-Fold Cross-Validation

- Mechanism:
    - Divides the dataset into K folds (commonly K=5 or 10).
    - Ensures that each fold preserves the same class distribution as the original dataset.
    - Each fold is used once as a validation set while the remaining K-1 folds are used for training.
- Justification:
    - Guarantees that minority class samples are present in every fold.
    - Prevents training or validation on folds with few or no fraud cases.
    - Reduces variance and provides a more stable estimate of model performance.

2. Optional Enhancements

1. Repeated Stratified K-Fold:
    - Repeats stratified K-Fold multiple times with different random splits.
    - Further reduces variance and increases reliability of evaluation.

	2. Use of Appropriate Metrics: ○ Evaluate model with metrics suitable for imbalanced data, such as Precision, Recall, F1-score, AUC-ROC, or MCC, instead of accuracy. 3. Integration with Resampling Techniques (if necessary): ○ Combine with SMOTE (Synthetic Minority Oversampling Technique) or undersampling within each fold to improve model learning of minority class patterns. Illustrative Example ● Dataset: 100,000 transactions, 1% fraud. ● Stratified 5-Fold CV: Each fold contains 20,000 transactions, preserving the 1% fraud proportion (~200 fraud cases per fold). ● During each iteration: ○ Train on 4 folds (~80,000 transactions, ~800 frauds). ○ Validate on 1 fold (~20,000 transactions, ~200 frauds). ● Aggregate performance metrics (e.g., F1-score, AUC-ROC) across folds for a robust estimate.
2	A movie recommender system is evaluated using only precision and recall. Propose a composite metric that also considers user satisfaction. Justify why your metric is more effective than traditional measures. Traditional evaluation metrics like precision and recall measure relevance of recommendations but do not capture user satisfaction, such as diversity, novelty, or coverage of recommended

items. A recommender system may achieve high precision by suggesting popular movies repeatedly, which may not satisfy users seeking variety or new content. Therefore, a composite metric is needed to incorporate relevance and user satisfaction.

Proposed Composite Metric: Weighted F1-Score with Diversity (WF1-D)

$$WF1\text{-}D = \alpha \cdot F1 + (1 - \alpha) \cdot Diversity$$

- **F1:** Harmonic mean of precision and recall, ensuring relevance of recommended items.
- **Diversity:** Measures how different the recommended movies are from each other. Can be computed using genre, director, or content-based dissimilarity:

$$Diversity = \frac{2}{N(N-1)} \sum_{i<j} (1 - sim(i, j))$$

where $sim(i, j)$ is the similarity between items i and j, and N is the number of recommended items.

- **α:** Weighting factor (0 ≤ α ≤ 1) to balance relevance vs diversity (user satisfaction).

Justification of Effectiveness

1. **Captures User Satisfaction:**
    - Encourages recommending **novel and varied movies**, preventing redundancy and improving engagement.
    - Users are more likely to explore and return to the platform when recommendations are diverse.
2. **Balances Relevance and Satisfaction:**
    - Precision and recall ensure that recommended items are relevant.
    - Diversity component prevents recommendations from focusing solely on popular items.
3. **Customizable for Business Goals:**
    - α can be tuned depending on the application—e.g., streaming platforms may

prioritize **diversity for engagement** or **precision for subscription retention**.

Example Application

- Suppose a recommender suggests 10 movies to a user:
    - 7 relevant (precision=0.7, recall=0.7)
    - Among these, 6 are of the same genre (low diversity ~0.2)
- Weighted metric with α=0.7:

$$WF1\text{-}D = 0.7 \cdot 0.7 + 0.3 \cdot 0.2 = 0.49 + 0.06 = 0.55$$

- Shows that even though relevance is moderate, **low diversity penalizes overall score**, reflecting lower user satisfaction.

Metric	Focus	Limitation	Advantage of WF1-D
Precision	Relevance of recommended items	Ignores diversity and novelty	Considers relevance + user satisfaction
Recall	Coverage of relevant items	Ignores redundancy and novelty	Encourages coverage while rewarding diverse recommendations
F1	Balances precision and recall	Still ignores user satisfaction	Combined with diversity to improve engagement and satisfaction

Using WF1-D (Weighted F1 with Diversity) as a composite metric allows the evaluation of recommender systems in terms of both accuracy and user satisfaction. Unlike traditional precision and recall, it penalizes redundant or overly similar recommendations, promoting novelty and engagement, which are critical for retaining users and improving long-term system effectiveness. By adjusting the weight α, the metric can be tailored to different business or user goals, making it a more holistic and practical evaluation measure.
