Group 2(ML Unit 2 practice questions)

Sequence of questions solved by the students (2 questions per head)

1. Ritesh Borikar
2. Sahil Sharma
3. Puruvas Pathak
4. Piyush Pancheshwar
5. Rahul Mohurley
6. Kaveri Ganar
7. Priya Dhole
8. Saloni Thakur
9. Vibhanshu Mandagode
10. Nitiksha Gupta
11. Meena Surjose

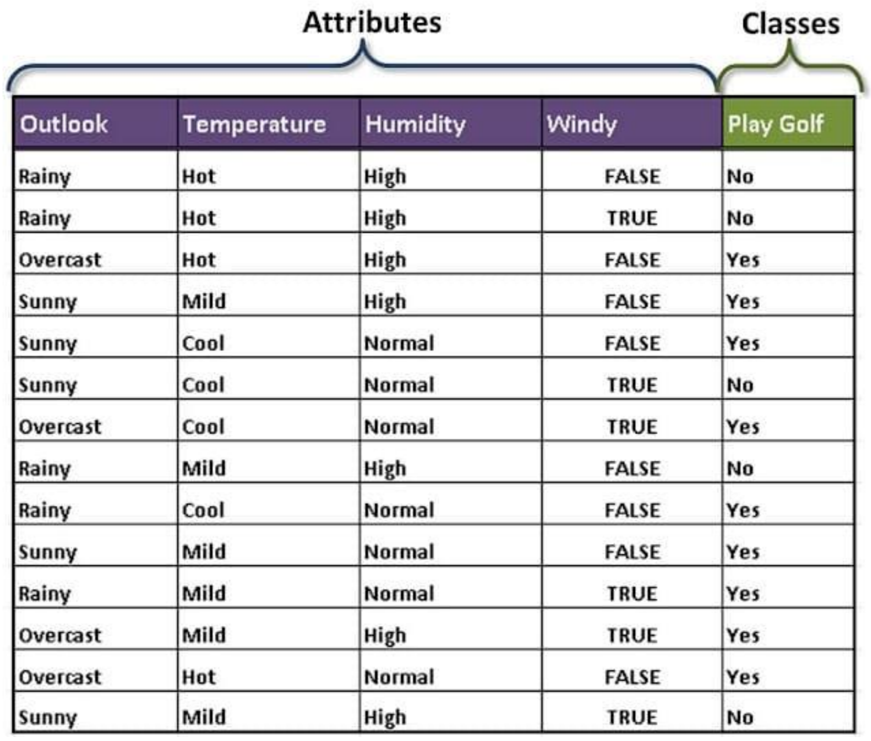
1) Describe the properties of a rule-based classifier?

Ans: - A rule-based classifier is a type of classification algorithm that operates by applying a set of rules to input data in order to assign them to one or more predefined categories or classes. These classifiers are often used in decision-making systems where the decision process can be easily articulated in the form of rules. Here are some key properties of rule-based classifiers:

1. **Interpretability**: Rule-based classifiers are highly interpretable since they directly represent the decision logic in the form of rules. Each rule typically consists of a condition (or set of conditions) and an associated class label or decision.
2. **Transparency**: Due to their simple and explicit nature, rule-based classifiers offer transparency in decision-making. It is easy to understand why a particular decision was made by inspecting the rules applied.
3. **Ease of Implementation**: Rule-based classifiers are relatively easy to implement since they do not require complex mathematical optimization procedures. The rules can often be defined manually by domain experts based on their knowledge of the problem domain.
4. **Human Understandability**: Because of their transparent nature, rule-based classifiers are easy for humans to understand and interpret, making them suitable for applications where human experts need to validate or refine the decision-making process.

5. **Scalability**: Rule-based classifiers can handle large datasets efficiently, especially if the rules are properly structured and optimized. However, as the number of rules increases, managing and optimizing them can become challenging.
6. **Accuracy**: The accuracy of rule-based classifiers depends on the quality of the rules defined. If the rules accurately capture the underlying patterns in the data, the classifier can achieve high accuracy. However, if the rules are too simplistic or fail to capture the complexity of the data, the classifier's performance may suffer.
7. **Robustness**: Rule-based classifiers can be robust to noisy data since they focus on capturing general patterns rather than fitting specific data points. However, they may struggle with highly imbalanced datasets or complex data distributions where defining accurate rules is challenging.
8. **Adaptability**: Rule-based classifiers can be easily updated or modified by adding, removing, or modifying rules based on new data or changes in the problem domain. This adaptability makes them suitable for dynamic environments where the decision criteria may evolve over time.

2) Analyze outlook attribute from given dataset using Decision Tree.

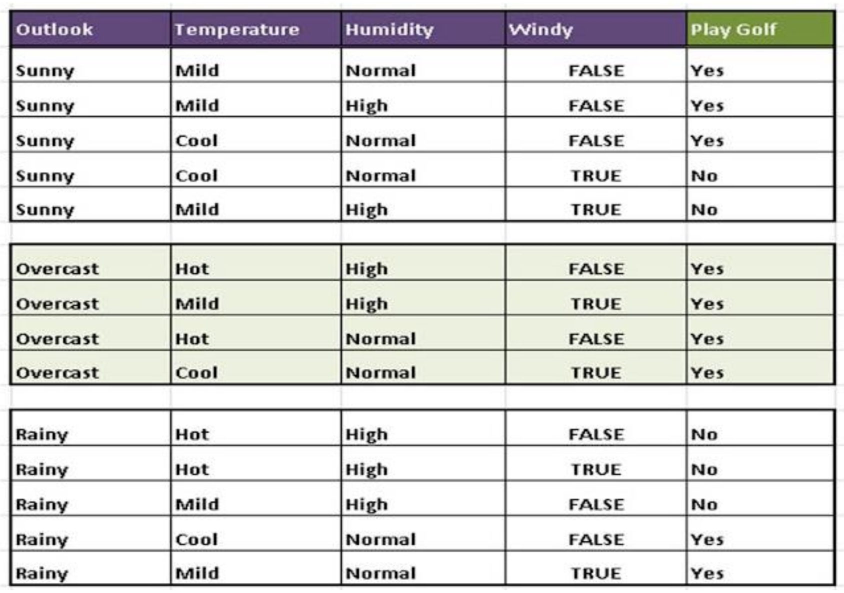


Attributes				Classes
Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Ans: - Step by Step Procedure for Building a Decision Tree

Step 1: Determine the Decision Column

Bifurcate or split the outlook attribute on the bases of 3 values i.e. Sunny, overcast and Rainy.



Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Table 1 initial split using Outlook

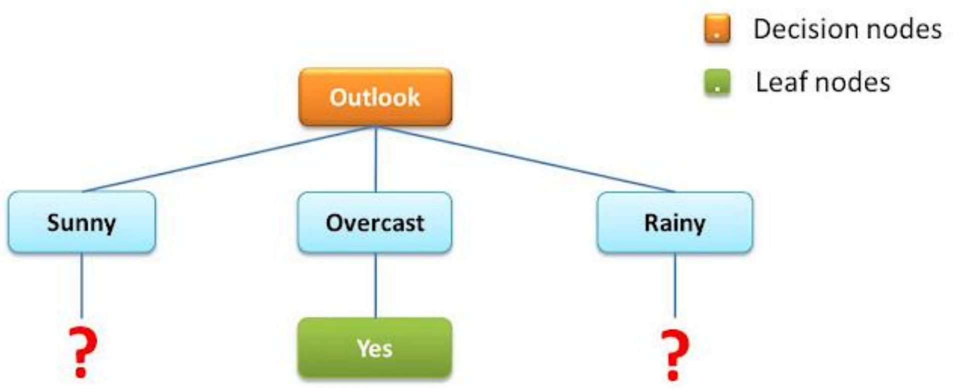
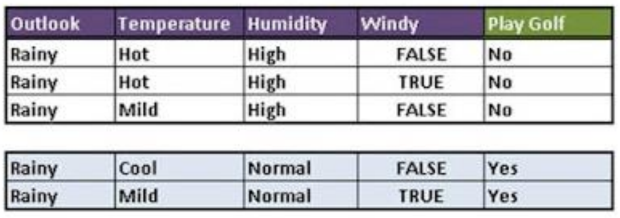


Fig.1 Decision Tree after 1st Split

Step 2: - Split the Rainy Outlook's using Humidity Attribute.



Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Table 2 Split using Humidity

From above table, we can say that whenever there is **Rainy** outlook with **High Humidity**, we can't **play Golf**. When the there is **Rainy** outlook with **Normal Humidity**, we can **play Golf**

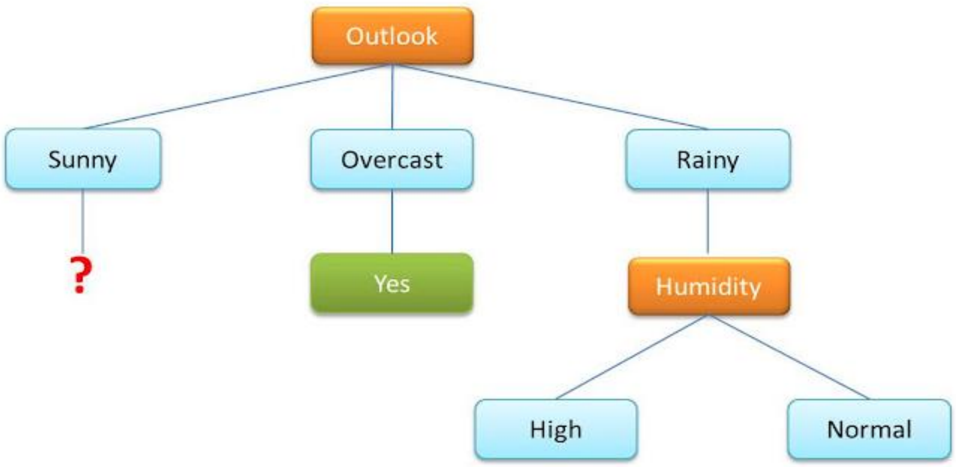
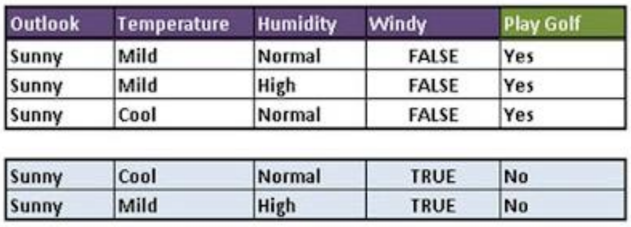


Fig. 2 Split using Humidity Attribute

Step 3: - Split Sunny outlook using Windy attribute



Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No

Table 3 Split using windy attribute

From the above table, we can say that whenever there is **Sunny Outlook** with No Wind or **FALSE Windy** condition then we can take the decision to **Play Golf**. When there is **Sunny outlook** with **TURE Windy** condition, we can take the decision of **NOT** to **Play Golf**

Step 4: - Make the Final Decision Tree

Make the Final Decision Tree by considering all the above split table.

Fig. 3 Final Decision Tree

3) **How does instance -based learning work? Explain with an example?**

Ans: - Instance-based learning, also known as instance-based learning or memory-based learning, is a type of machine learning algorithm that learns from examples stored in memory. Instead of generalizing from a training dataset to create a model, instance-based learning algorithms make predictions by comparing new input instances to instances seen during training.

Here's how instance-based learning typically works:

- **Training Phase:** During the training phase, the algorithm stores the training instances in memory without any explicit model-building process. Each training instance is associated with its corresponding label or output.
- **Prediction Phase:** When a new instance is presented for prediction, the algorithm compares it to the instances stored in memory and identifies the most similar instances. -
- **Decision Making:** Instance-based learning algorithms typically use a measure of similarity (e.g., Euclidean distance, cosine similarity) to determine the closeness between the new instance and the stored instances. The algorithm may then use a `RFC voting scheme or a weighted average of the labels of the most similar instances to make a prediction for the new instance.
- **Updating Memory:** Some instance-based learning algorithms may update their memory with new instances as they are encountered during the prediction phase. This allows the algorithm to adapt to new data without retraining the entire model.

Example:

Let's say we have a dataset of iris flowers with features such as sepal length, sepal width, petal length, and petal width, along with their corresponding species labels (e.g., setosa, versicolor, virginica). We want to build an instance-based learning algorithm to predict the species of a new iris flower based on its features.

- **Training Phase:** We store the features and labels of the iris flowers in memory.
- **Prediction Phase:** When a new iris flower is presented with its features, the algorithm computes the similarity between the new instance and all the instances in memory. It may use a distance metric like Euclidean distance.

- **Decision Making:** The algorithm identifies the k most similar instances (neighbors) to the new instance. For example, if we use a k-nearest neighbors (KNN) algorithm with k=3, we find the three iris flowers in the dataset that are most similar to the new instance.
- **Voting Scheme:** If we're using a voting scheme, the algorithm counts the number of instances of each species among the k nearest neighbors and predicts the species with the highest count.
- **Weighted Average:** Alternatively, we may assign weights to the neighbors based on their similarity to the new instance and compute a weighted average of the labels to make a prediction.
- **Updating Memory:** If the instance-based learning algorithm supports it, it may update its memory with the features and label of the new instance for future predictions.

Instance-based learning is particularly useful when the relationship between inputs and outputs is complex or not easily characterized by a simple model. It can also handle noisy data well and adapt to changes in the data distribution over time. However, instance-based learning algorithms may be computationally expensive, especially with large datasets, since they require storing and comparing instances during prediction.

4) Elaborate the term instance-based learning?

Ans: -The Machine Learning systems which are categorized as instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure. It is called instance-based because it builds the hypotheses from the training instances. It is also known as memory-based learning or lazy-learning (because they delay processing until a new instance must be classified). The time complexity of this algorithm depends upon the size of training data. Each time whenever a new query is encountered, its previously stores data is examined. And assign to a target function value for the new instance.

The worst-case time complexity of this algorithm is O (n), where n is the number of training instances. For example, If we were to create a spam filter with an instance-based learning algorithm, instead of just flagging emails that are already marked as spam emails, our spam filter would be programmed to also flag emails that are very similar to them. This requires a measure of resemblance between two emails. A similarity measure between two emails could be the same sender or the repetitive use of the same keywords or something else.

Advantages:

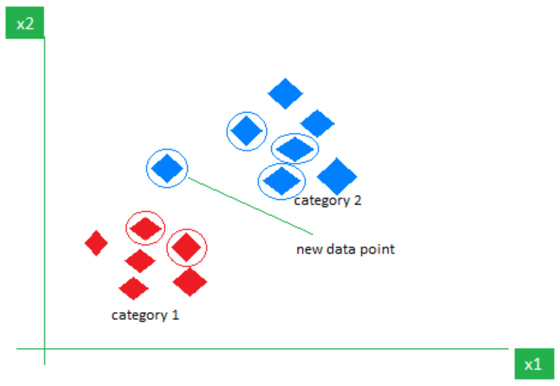
1. Instead of estimating for the entire instance set, local approximations can be made to the target function.
2. This algorithm can adapt to new data easily, one which is collected as we go .

Disadvantages:

1. Classification costs are high
2. Large amount of memory required to store the data, and each query involves starting the identification of a local model from scratch.

Some of the instance-based learning algorithms are :

1. **K Nearest Neighbor (KNN): -** The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.



2. **Self-Organizing Map (SOM): -Self Organizing Map (or Kohonen Map or SOM)** is a type of Artificial Neural Network which is also inspired by biological models of neural systems from the 1970s. It follows an unsupervised learning approach and trained its network through a competitive learning algorithm. SOM is used for clustering and mapping (or dimensionality reduction) techniques to map multidimensional data onto lower-dimensional which allows people to reduce complex problems for easy interpretation. SOM has two layers, one is the Input layer and the other one is the Output layer.
3. **Learning Vector Quantization (LVQ): - Learning Vector Quantization (or LVQ)** is a type of Artificial Neural Network which also inspired by biological models of neural systems. It is based on prototype supervised learning classification algorithm and trained its network through a competitive learning algorithm similar to Self

Organizing Map. It can also deal with the multiclass classification problem. LVQ has two layers, one is the Input layer and the other one is the Output layer.

4. **Locally Weighted Learning (LWL): -** Locally weighted learning (LWL) Locally Weighted Learning methods are a class of function approximation techniques, where a local model is created for each point of interest based on neighbouring data, instead of building a global model for the whole function space. LWL methods are non-parametric.
5. **Case-Based Reasoning: -** Case-based reasoning (CBR) is a paradigm of artificial intelligence and cognitive science that models the reasoning process as primarily memory based. Case-based reasoners solve new problems by retrieving stored 'cases' describing similar prior problem-solving episodes and adapting their solutions to fit new needs.

5) Analyze the working of decision tree learning? How it is helpful in association with machine learning?

Ans - A decision tree is a flowchart-like structure where an internal node represents a "test" on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

example:

Let's say we want to build a decision tree to classify whether a person plays tennis on a particular day based on weather conditions. The attributes (features) we consider are Outlook, Temperature, Humidity, and Wind.

Outlook: Sunny, Overcast, Rainy

Temperature: Hot, Mild, Cool

Humidity: High, Normal

Wind: Weak, Strong

```
              Outlook
          /      |      \
       Sunny  Overcast  Rainy
        |          |      |
    Humidity?     Yes   Wind?
    /        \                    /   \
High   Normal     Weak   Strong
   |           |                  |       |
Yes     No                  Yes     No
```

Here, each internal node represents a decision based on a feature, and each leaf node represents a class label (in this case, "Yes" for playing tennis and "No" for not playing tennis). For example, if the Outlook is Sunny and Humidity is High, the decision tree predicts "No" (since it leads to a leaf node labeled "No").

Decision trees are easy to interpret and can handle both numerical and categorical data. They can handle missing values and are robust to outliers.
decision tree in association with machine learning:

- Decision trees produce models that are easy to interpret and understand.
- Decision trees can handle missing values in the dataset without requiring imputation techniques.
- Decision trees serve as the building blocks for ensemble methods like Random Forests and Gradient Boosting Machines.

6) Using Random Forest how absolute results can be obtained? Justify your answer with the help of example.
Ans - Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive accuracy and robustness. It works by constructing a multitude of decision trees during training and outputting the mode of the classes (classification) or the average prediction (regression) of the individual trees.
Random Forest can provide more reliable and accurate results compared to a single decision tree due to the following reasons:

- By averaging the predictions of multiple trees, Random Forest reduces the tendency to overfit the training data.
- The ensemble nature of Random Forest helps in improving the generalization performance of the model.

- Random Forest can also provide insights into feature importance, which can be useful for feature selection and understanding the underlying patterns in the data.

Example:

Suppose you have a dataset containing information about various characteristics of houses (e.g., size, number of bedrooms, location) and their corresponding sale prices. You want to predict the sale price of a house based on its features.

You decide to use Random Forest for this task. You construct an ensemble of decision trees, each trained on a random subset of the training data and a random subset of the features. During prediction, you aggregate the predictions of all trees to obtain the final result.

For instance, if you have a Random Forest consisting of 100 decision trees, and you want to predict the sale price of a new house, you pass the features of the house through each of the 100 trees. Each tree will make its own prediction based on its subset of data and features. Finally, you take the average of all 100 predictions to obtain the final predicted sale price.

7) Analyze different challenges and applications of cased based learning.

Ans - Case-based learning (CBL) in machine learning (ML) offers a unique approach to training models by using real-world cases or examples for analysis, pattern recognition, and decision-making. Here's an analysis of the challenges and applications specific to CBL in ML:

<u>**Challenges: -**</u>

1. Data Quality and Availability: One of the primary challenges in CBL for ML is obtaining high-quality and relevant datasets. Real-world cases may suffer from data inconsistencies, noise, bias, or missing values, which can affect the performance of ML models trained on them.

2. Feature Engineering: CBL often requires careful feature engineering to represent cases effectively. Identifying relevant features and transforming raw data into a suitable format for ML algorithms can be challenging, especially for complex, unstructured data types such as text or images.

3. Model Complexity and Generalization: ML models trained using CBL may struggle with overfitting or lack of generalization if the cases used for training are too specific or limited in scope. Balancing model complexity with generalizability is essential to ensure robust performance on unseen data.

4. Interpretability and Explainability: CBL may result in ML models that are difficult to interpret or explain, particularly when using complex algorithms such as deep learning. Understanding how models make predictions based on the underlying cases is crucial for building trust and transparency in ML systems.

5. Computational Resources: Training ML models on large-scale case datasets can be computationally intensive and may require significant resources, including computational power, storage, and memory. Managing and scaling infrastructure to accommodate the computational demands of CBL can pose challenges, particularly for resource-constrained environments.

Applications: -

1. Predictive Analytics: CBL can be applied to predictive modeling tasks, where ML models learn from historical cases to make predictions about future outcomes. Examples include predicting customer churn, forecasting stock prices, or diagnosing medical conditions based on patient data.

2. Anomaly Detection: CBL is well-suited for anomaly detection applications, where ML models learn to identify unusual patterns or outliers in data. By analyzing past cases, models can detect anomalies that deviate from normal behavior, such as fraudulent transactions or network intrusions.

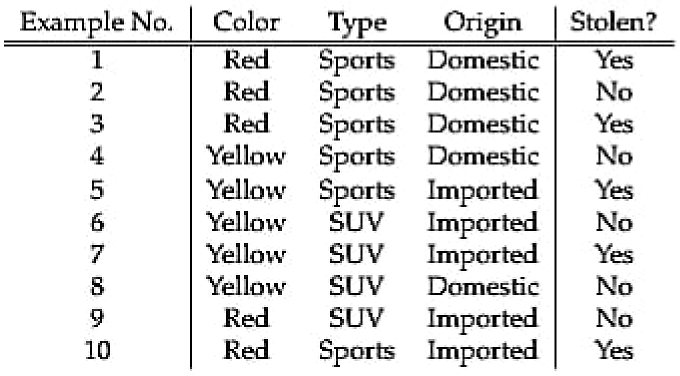
3. Recommendation Systems: CBL can power recommendation systems by learning from past user interactions or preferences to provide personalized recommendations. By analyzing historical cases of user-item interactions, ML models can suggest relevant products, movies, or articles to users.

4. Natural Language Processing (NLP): In NLP applications, CBL can be used to train ML models for tasks such as text classification, sentiment analysis, or named entity recognition. By learning from annotated text data or historical examples, models can generalize patterns and structures in language.

5. Healthcare Decision Support: CBL has applications in healthcare decision support systems, where ML models learn from past patient cases to assist clinicians in diagnosis, treatment planning, and prognosis prediction. By analyzing electronic health records, medical imaging data, or genomic data, models can provide valuable insights and recommendations to healthcare providers.

In summary, while CBL in ML offers opportunities for building predictive models, detecting anomalies, powering recommendation systems, and assisting in decision-making across various domains, it also presents challenges related to data quality, feature engineering, model complexity, interpretability, and resource requirements. By addressing these challenges and leveraging its applications effectively, CBL can contribute to the development of robust and reliable ML solutions.

8) Analyze Color attribute from given dataset using Decision Tree.



Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Ans -

9) With the help of suitable example explain how rule-based system works.

Ans –

A rule-based system in AI generates an output by using a collection of inputs and a set of rules. The system first determines which principles apply to the inputs. If a rule is applicable, the system executes the corresponding steps to generate the output. If no guideline is applicable, the system might generate a default output or ask the user for more details.

a rule-based system in AI consists of seven fundamental elements:

1. **The knowledge base**:

It contains the specialized expertise required for problem-solving. The information is represented as a set of rules in a rules-based system. Every rule has an IF (condition) THEN (action) structure and defines a relationship, suggestion, directive, strategy, or heuristic. The rule is activated, and the action portion is carried out as soon as the conditional portion of the rule is met.

2. **The database**:

The database contains a collection of facts compared to the knowledge base's rules IF (condition) clause.

3. **The inference engine**:

The expert system uses the inference engine to derive the logic and arrive at a conclusion. The inference engine's task is to connect the facts kept in the database with the rules specified in the knowledge base. The semantic reasoner is another name for the reasoning engine. It deduces information or executes necessary actions based on data and the rule base present in the knowledge base.

4. **Explanations facilities**:

The user can use the explanation facilities to question the expert system on how it came to a particular conclusion or why a particular fact is necessary. The expert system must be able to defend its logic, recommendations, analyses, and conclusions.

5. **User Interface**:

The user interface is the channel through which the user interacts with the expert system to find a solution to an issue. The user interface should be as simple and intuitive as possible, and the dialogue should be as helpful and friendly as possible.

6. **External connection**:

An expert system can interact with external data files and programs written in traditional computer languages like C, Pascal, FORTRAN, and Basic, thanks to the external interface.

7. **Active recall**:

The working memory keeps track of transient data and knowledge.

Examples of Rule-based Systems

Healthcare, finance, and engineering are just a few examples of the sectors and applications that use rule-based systems. Following are some instances of a rule-based system in AI:

1. **Medical Diagnosis:**

Based on a patient's symptoms, medical history, and test findings, a rule-based system in AI can make a diagnosis. The system can make a diagnosis by adhering to a series of guidelines developed by medical professionals.

2. **Fraud Detection:**

Based on particular criteria, such as the transaction's value, location, and time of day, a rule-based system in AI can be used to spot fraudulent transactions. The system, for the additional examination, can then flag the transaction.

3. **Quality Control:**

A rule-based system in AI can ensure that products satisfy particular quality standards. Based on a set of guidelines developed by quality experts, the system can check for flaws.

4. **Decision support systems:**

They are created to aid decision-making, such as choosing which assets to buy or what to buy.

10) Construct Rule Based Learning Model with suitable example

Ans –

11) What are the key differences between K-NN and K-means algorithms?

Ans -

Feature	**K-Nearest Neighbours (KNN)**	**K-Means**
Type of algorithm	Supervised learning	Unsupervised learning
Purpose	Classification or regression	Clustering
Main idea	Classify data points based on similarity	Group data points into 'k' clusters
Training data	Requires labeled training data	Requires unlabeled training data
Decision making	Based on the majority vote of nearest neighbors	Based on centroids of clusters

Distance measure	Typically Euclidean distance, but can vary	Typically Euclidean distance, but can vary
Parameter tuning	Requires tuning of k (number of neighbors)	Requires tuning of k (number of clusters)
Computational complexity	Higher complexity due to searching for nearest neighbors	Lower complexity compared to K-NN
Outlier sensitivity	Sensitive to outliers	Less sensitive to outliers
Scalability	Scales poorly with large datasets and dimensions	More scalable for large datasets and dimensions
Use cases	Classification, regression	Clustering, data preprocessing

12) Compare & Contrast Instance-based & Model-based Learning?

Ans -

Feature	**Instance-Based Learning**	**Model-Based Learning**
Definition	Makes predictions based on similarities between new instances and training instances	Constructs an explicit model from the training data for making predictions
Representation	Stores the entire training dataset	Constructs a model capturing underlying patterns in the data
Learning Process	Memorizes training data	Learns from training data to find optimal model parameters or structure

Generalization	Tends to have low generalization capabilities as it memorizes training data	Generally, has better generalization capabilities as it learns underlying patterns
Computational Cost	Low during training, potentially high during prediction	Can be higher during training, but usually lower during prediction
Memory Requirement	High, requires storing the entire training dataset	Lower, only requires storing model parameters or structure

13)How are the different nodes of decision trees represented?

OR

14)Design and explain the structure of a Decision Tree?

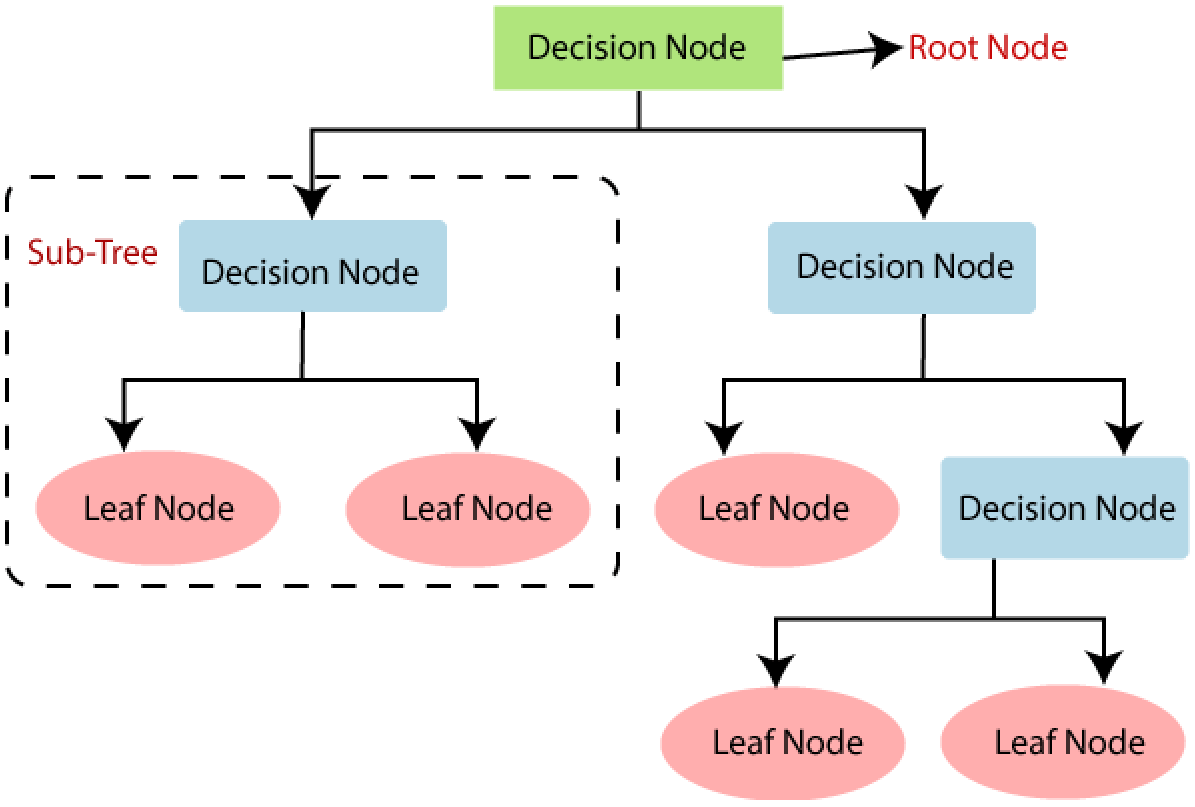
Ans - A decision tree is a hierarchical structure that consists of nodes connected by edges. Here's a detailed breakdown of the different types of nodes typically found in decision trees:

Root Node: This is the topmost node of the tree, representing the initial decision or starting point. It contains the entire dataset and is split into child nodes based on the feature that best separates the data according to a specific criterion (e.g., Gini impurity, information gain).

Leaf Nodes (Terminal Nodes): Leaf nodes are the endpoints of the decision tree. They do not contain any further splits and represent the final outcome or decision. Each leaf node typically corresponds to a class label (for classification tasks) or a continuous value (for regression tasks). When a data sample reaches a leaf node, it is assigned the class label or value associated with that node.

Parent and Child Nodes: In a decision tree, nodes are organized hierarchically, with parent nodes splitting into child nodes. The root node is the parent of all other nodes, and each internal node is a parent to its child nodes. Child nodes are the nodes that result from splitting the parent node based on a specific feature and threshold.

Tree Pruning: Pruning is an optional step to reduce the size of the decision tree. It involves removing unnecessary nodes or branches that do not contribute significantly to the predictive accuracy of the tree. Pruning helps prevent overfitting and improves the generalization ability of the decision tree model.



15) Suppose you are working on a dataset with imbalanced classes. What strategies can you use in ensemble learning to handle this problem?

Ans: -

- To handle imbalanced classes, we can use data-level techniques such as Oversampling, Undersampling
    1. **Oversampling:** Replicate instances from the minority class to balance the class distribution. Techniques like SMOTE (Synthetic Minority Oversampling Technique) create synthetic data points to avoid overfitting.
    2. **Undersampling:** Randomly remove instances from the majority class to match the size of the minority class. This is faster but can discard valuable information

- Additionally, one could use a cost-sensitive learning approach that assigns different weights to different classes to account for the imbalance.
    1. **Cost-Sensitive Learning:** Assign higher weights to misclassified minority class instances during training, forcing the model to focus on learning these rarer examples.
    2. **Threshold Tuning:** Adjust decision thresholds in classifiers to prioritize minority class accuracy over majority class accuracy.

- Another approach is to use ensemble learning techniques such as bagging or boosting to improve the performance of minority class predictions.
    1. **Boosting:** Algorithms like AdaBoost and XGBoost focus on learning from previously misclassified instances, iteratively improving performance on the minority class.
    2. **Bagging:** Techniques like Random Forests create multiple independent classifiers trained on different subsets of the data. This diversity can handle imbalanced data effectively.

16) How is a Random Forest related to Decision Trees with an example?

Ans: - A Random Forest is a type of ensemble learning method that consists of a collection of decision trees. Each tree in the Random Forest is trained independently on a random subset of the data and a random subset of the features. The final prediction of the Random Forest is typically obtained by averaging or voting over the predictions of all individual trees.

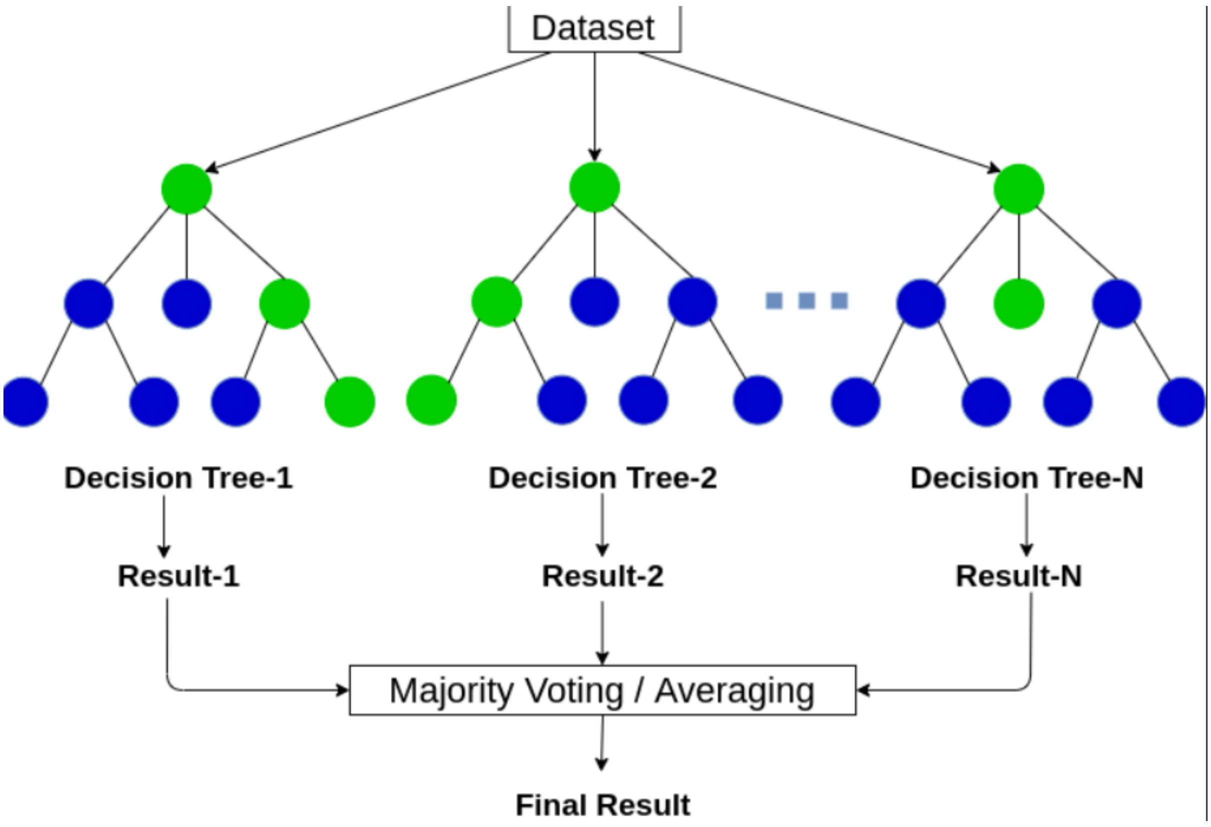
Here's an example to illustrate the relationship:

Suppose you have a dataset with information about customers, including features like age, income, and buying habits. You want to predict whether a customer will make a purchase or not. You decide to use both Decision Trees and Random Forest for this task.

Decision Trees: You train a decision tree on the dataset, where the tree learns to split the data based on the features (e.g., age < 30, income > $50,000). The decision tree makes predictions by following a series of if-else rules based on these splits.

Random Forest: Instead of relying on a single decision tree, you build a Random Forest consisting of multiple decision trees. Each tree is trained on a random subset of the data and features. When making predictions, each tree in the forest independently predicts whether the customer will make a purchase or not. The final prediction of the Random Forest is obtained by averaging (for regression) or voting (for classification) over the predictions of all trees.

In summary, a Random Forest is related to Decision Trees in that it is composed of multiple decision trees, but it incorporates randomness in both the data and feature selection process to improve robustness and generalization.



17) Suppose you have a dataset with many features and you're trying to predict a binary outcome. What type of ensemble learning algorithm would you recommend and why?

Ans - For predicting a binary outcome with a dataset containing many features, a good choice for an ensemble learning algorithm would be the Random Forest. Random Forest is a powerful and versatile ensemble method that combines multiple decision trees to make predictions. Here are some reasons why Random Forest might be a suitable choice:

Random Forest:-

• **High Predictive Accuracy:** Aggregates predictions from multiple decision trees to reduce overfitting and improve accuracy.

• **Handles Large Feature Sets:** Can handle datasets with many features without feature selection or dimensionality reduction.

• **Robust to Outliers**: Averages predictions across multiple trees, making it robust to outliers and noisy data.

• **Reduced Overfitting**: Uses bagging to train each tree on a random subset of data, reducing the risk of overfitting.

• **Automatic Feature Selection:** Evaluates feature importance during training, aiding in feature selection.

• **Parallelizable Training:** Can be trained in parallel, making it computationally efficient for large datasets.

18) Why do we use rule-based learning?

Ans - Rule-based learning is used for several reasons: -

Interpretability: Rule-based models are inherently interpretable. The rules are often in the form of "if-then" statements, making it easy to understand and explain the decision-making process to stakeholders, users, or regulatory bodies.

Transparency and Trust: The transparency of rule-based models fosters trust among users and stakeholders. Knowing the rules behind the model's predictions allows for better acceptance and understanding of the model's behavior.

Domain Expertise Integration: Rule-based systems can incorporate domain expertise directly into the model. Experts can contribute their knowledge by specifying rules that reflect their understanding of the problem, leading to more accurate and meaningful predictions.

Explicit Representation of Knowledge: Rule-based systems explicitly represent knowledge in a structured format. This makes it easier to encode and incorporate existing knowledge or business rules into the model, ensuring that the model aligns with domain-specific requirements.

Handling Missing Data and Uncertainty: Rule-based models can be designed to handle missing data or uncertain situations effectively. Rules can be crafted to make decisions even when some information is not available, providing robustness in real-world scenarios.

Ease of Maintenance: Rule-based systems are often modular and easier to maintain. Adding, modifying, or removing rules can be done without affecting the entire model, making it adaptable to changing requirements and evolving datasets.

Rule Extraction from Data: In some cases, rule-based models can automatically extract rules from data, providing insights into the underlying patterns in the dataset. This can be valuable for understanding the factors influencing predictions.

Small to Medium-sized Datasets: Rule-based learning can be particularly effective for small to medium-sized datasets where the explicit representation of rules can lead to accurate predictions without the need for massive amounts of training data.
Regulatory Compliance: In regulated industries, where transparency and interpretability are crucial, rule-based models can help meet regulatory compliance requirements by providing clear explanations for decision-making processes.

19) Distinguish between Random Forest and Decision Trees with an example.

Ans - Decision Tree: Decision Tree is a supervised learning algorithm used in machine learning. It operated in both classification and regression algorithms. As the name suggests, it is like a tree with nodes. The branches depend on the number of criteria. It splits data into branches till it achieves a threshold unit. A decision tree has root nodes, children nodes, and leaf nodes. Recursion is used for traversing through the nodes. It handles data accurately and works best for a linear pattern.

Random Forest: Random Forest is yet another very popular supervised machine learning algorithm that is used in classification and regression problems. One of the main features of this algorithm is that it can handle a dataset that contains continuous variables, in the case of regression. Simultaneously, it can also handle datasets containing categorical variables, in the case of classification. This in turn helps to deliver better results for classification problems. The basic difference being it does not rely on a singular decision. It assembles randomized decisions based on several decisions and makes the final decision based on the majority. You can infer Random forest to be a collection of multiple decision trees.

Feature	**Decision Tree**	**Random Forest**
Algorithm Type:	Decision Tree is a single algorithm that constructs a tree-based model.	Random Forest is an ensemble algorithm that combines multiple decision trees.
Independence:	Decision Trees are built independently.	Random Forest builds each tree independently and combines their outputs through averaging

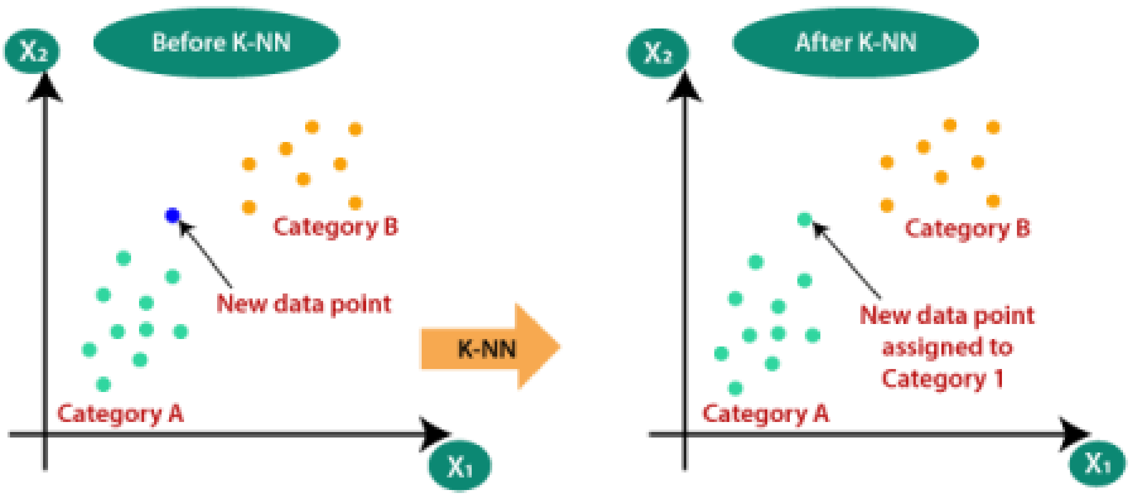
		(regression) or voting (classification).
Overfitting:	Decision Trees are prone to overfitting, especially when the tree is deep and not pruned.	Random Forest reduces overfitting by averaging or voting among multiple trees.
Prediction Speed:	Decision Trees are faster to train but slower to predict, especially with deep trees.	Random Forest is slower to train due to building multiple trees but faster to predict as predictions are parallelizable.
Interpretability:	Decision Trees are easily interpretable and allow understanding of decision-making processes.	Random Forest is less interpretable due to the complexity introduced by multiple trees.
Scalability:	Decision Trees are suitable for small to medium-sized datasets	Random Forest is more scalable and can handle larger datasets efficiently by parallelizing tree construction.
Example:	Imagine sorting fruits based on their color: if it's red, it's an apple; if it's orange, it's an orange.	Instead of asking one person about the weather, you ask a group of people and decide based on the majority opinion.

20) Classify K-NN and K-means algorithms based on their performances.

Ans: – KNN: -

1. KNN, or K-Nearest Neighbours, is a non-parametric classification algorithm that is used to identify patterns in data.
2. The KNN algorithm works by finding the K closest data points to the input, or query, point. These data points labels are then used to classify the query point.

3. The number of neighbours, K, is a hyperparameter that is usually chosen before running the algorithm.
4. KNN is straightforward to implement and can work well for smaller datasets.
5. It is commonly used for predicting outcomes based on similar inputs.
6. One of the advantages of KNN is that it is a lazy learning algorithm, meaning that it does not require any training data to be processed before making predictions.
7. This makes it a useful algorithm for real-time applications where data is constantly changing.
8. For example:
    I. KNN can be used to predict a customer's buying behaviour based on their past purchases.
    II. KNN can also be used to diagnose diseases based on patient data or recognize faces in images.
    III. Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.



K-means:

1. K-Means is a clustering algorithm that groups similar data points together.
2. The goal of K-Means is to separate data points into K clusters, where each data point belongs to the nearest cluster centre.

3. The centre of each cluster is determined by calculating the mean of all the points within the cluster.
4. K-Means is based on distance and works by minimizing the sum of squared distances between data points and their assigned cluster centres.
5. K-Means can be useful for pattern recognition tasks such as image segmentation or customer behaviour analysis.
6. One of the advantages of K-Means is its simplicity and efficiency. It is a fast algorithm that can handle large datasets with ease.
7. For example:
   I. K-Means can be used to group customers based on their purchasing behavior or segment a population based on demographic data.
   II. K-Means can also be used for image processing tasks such as image segmentation or object recognition.

References: -

- https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
- Knn Vs K-Means | Blog (playerzero.ai)
- https://kindsonthegenius.com/blog/how-to-build-a-decision-tree-for-classification-step-by-step-procedure-using-entropy-and-gain/
- https://medium.datadriveninvestor.com/decision-tree-algorithm-with-hands-on-example-e6c2afb40d38
- https://saturncloud.io/glossary/instancebased-learning/#:~:text=Instance%2Dbased%20learning%20is%20a,generalization%20process%20until%20prediction%20time
- https://www.geeksforgeeks.org/instance-based-learning/

- https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/#:~:text=A.,data%2C%20especially%20on%20regression%20tasks