**1. What types of Classification Algorithms do you know?**

Classification algorithms are used to categorize data into predefined classes or labels. Common types of classification algorithms include:

* **Logistic Regression**: A linear model used for binary classification problems.
* **Decision Trees**: A tree-like model used for making decisions based on feature values.
* **Random Forest**: An ensemble method using multiple decision trees to improve accuracy and reduce overfitting.
* **Support Vector Machines (SVM)**: A method that finds a hyperplane to separate data points of different classes.
* **K-Nearest Neighbors (KNN)**: A non-parametric method that classifies data points based on the majority class among the k-nearest data points.
* **Naive Bayes**: A probabilistic classifier based on Bayes' theorem with strong independence assumptions between features.
* **Neural Networks**: Deep learning models used for complex classification tasks with multiple layers of neurons.

**2. What are Decision Trees?**

* **Decision Trees** are a type of supervised learning algorithm used for both classification and regression tasks. The model predicts the target variable by learning simple decision rules inferred from the data features.
* A **Decision Tree** splits the data into smaller subsets based on feature values, forming a tree-like structure of decision nodes and leaf nodes.
* Each **internal node** represents a decision based on a feature, and each **leaf node** represents an output class (label).

**3. What type of node is considered Pure?**

* A node is considered **Pure** when all the data points in that node belong to the same class.
* In other words, if a node has samples of only one class (e.g., all 1's or all 0's), it is pure. This means the **impurity** or **entropy** of that node is zero.

**4. How are the different nodes of Decision Trees represented?**

* **Root Node**: The top-most node in a decision tree. It represents the entire dataset and is split based on the best feature.
* **Decision Nodes**: Internal nodes that represent decisions made based on feature values. These nodes split the data further into branches.
* **Leaf Nodes (Terminal Nodes)**: The end nodes of the decision tree that provide the final classification label. They represent the class labels (output) based on the splits made from the root to the leaf.

**5. What are some advantages of using Decision Trees?**

* **Easy to Interpret and Understand**: Decision trees provide a clear and visual representation of the decision-making process.
* **Handles Non-Linear Relationships**: They can capture complex non-linear relationships between features and the target variable.
* **Requires Little Data Preprocessing**: Decision trees do not require feature scaling or normalization, and they can handle both numerical and categorical data.
* **Feature Importance**: Decision trees help identify the most important features for prediction by analyzing the splits.

**6. What is Gini Index and how is it used in Decision Trees?**

* The **Gini Index** is a metric used to measure the impurity of a node in a decision tree. It represents the probability of a randomly chosen element being incorrectly classified if it was randomly labeled according to the distribution of classes in the node.
* **Formula for Gini Index**:

Gini Index=1−∑i=1npi2\text{Gini Index} = 1 - \sum\_{i=1}^{n} p\_i^2Gini Index=1−i=1∑n​pi2​

Where pip\_ipi​ is the probability of an element belonging to class iii.

* **Usage in Decision Trees**:
  + At each split in the decision tree, the algorithm evaluates the Gini index for each possible split to decide the best feature to split on.
  + The feature with the lowest Gini index is chosen, as it represents the purest split (lowest impurity).
  + In the provided code, the criterion="entropy" was used instead of Gini index, meaning **Information Gain** was the metric used. However, if criterion="gini" were specified, the Gini Index would be the measure for splitting.

**Possible Oral Questions and Answers**

1. **Q: What is the purpose of using train\_test\_split()?**
   * **A:** It is used to split the dataset into training and testing sets, allowing us to evaluate the model's performance on unseen data.
2. **Q: Why do we use the DecisionTreeClassifier in this code?**
   * **A:** The DecisionTreeClassifier is used to classify data based on decision rules derived from the features, making it suitable for binary or multi-class classification tasks.
3. **Q: What does the criterion="entropy" parameter do in the Decision Tree model?**
   * **A:** It specifies that the model should use the entropy (information gain) method to determine the best splits at each node.
4. **Q: What is a confusion matrix?**
   * **A:** A confusion matrix is a table used to evaluate the performance of a classification model. It shows the counts of true positives, false positives, true negatives, and false negatives.
5. **Q: What are precision, recall, and F1-score in the classification\_report()?**
   * **A:**
     + **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.
     + **Recall**: The ratio of correctly predicted positive observations to all actual positives.
     + **F1-score**: The harmonic mean of precision and recall, used as a combined measure of accuracy.
6. **Q: What is the use of max\_depth in the Decision Tree model?**
   * **A:** The max\_depth parameter limits the depth of the tree to prevent overfitting and control the complexity of the model.
7. **Q: Why do we remove the 'Serial No.' column from the dataset?**
   * **A:** The 'Serial No.' column is not a predictive feature and does not affect the outcome, so it is removed to avoid unnecessary noise in the model.
8. **Q: What does an accuracy score of 1 mean?**
   * **A:** An accuracy score of 1 indicates a perfect prediction, where all test samples are correctly classified.

# Importing necessary library

import pandas as pd # Used for data manipulation and analysis

# Loading the dataset

df = pd.read\_csv('3.csv') # Reading the CSV file named '3.csv' into a DataFrame 'df'

# Displaying the columns of the DataFrame

df.columns # Returns the list of column names in the DataFrame

# Removing any trailing whitespace from column names

df.columns = df.columns.str.rstrip() # Strips any extra spaces at the end of column names

# Binarizing the 'Chance of Admit' column

df.loc[df['Chance of Admit'] >= 0.80, 'Chance of Admit'] = 1 # Setting 'Chance of Admit' to 1 if it is >= 0.80

df.loc[df['Chance of Admit'] < 0.80, 'Chance of Admit'] = 0 # Setting 'Chance of Admit' to 0 if it is < 0.80

# Displaying the modified 'Chance of Admit' column

df['Chance of Admit'] # Checking the transformed values (0 or 1)

# Checking updated columns again after changes

df.columns # Returns the updated list of column names

# Dropping the 'Serial No.' column as it is not useful for the prediction

df = df.drop('Serial No.', axis=1) # axis=1 means we are dropping a column

# Displaying the updated DataFrame

df # Shows the modified DataFrame after dropping the 'Serial No.' column

# Defining features (X) and target (Y) variables

x = df.iloc[:, 0:7] # Selecting the first 7 columns as features (independent variables)

y = df.iloc[:, 7] # Selecting the 8th column as the target variable ('Chance of Admit')

# Displaying the target variable

y # Outputs the 'Chance of Admit' column as Y

# Importing the function for splitting the data

from sklearn.model\_selection import train\_test\_split

# Splitting the data into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

# test\_size=0.25 means 25% of data is used for testing, and 75% for training

# Importing Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

# Creating and training the Decision Tree model

model = DecisionTreeClassifier(criterion="entropy", max\_depth=4)

# criterion="entropy" uses information gain to split nodes, max\_depth=4 limits the tree depth to 4 levels

# Fitting the model with training data

model.fit(x\_train, y\_train)

# Making predictions on the test data

y\_pred = model.predict(x\_test) # Predicting the target variable using the test data

# Displaying the predicted values

y\_pred # Output the predicted class labels (0 or 1)

# Importing the confusion matrix function

from sklearn.metrics import confusion\_matrix

# Generating the confusion matrix

matrix = confusion\_matrix(y\_test, y\_pred, labels=[0.0, 1.0])

# Confusion matrix helps evaluate the performance of the classification model

# Checking the shapes of training and testing datasets

x\_train.shape # Outputs the shape of x\_train (number of rows, number of features)

x\_test.shape # Outputs the shape of x\_test (number of rows, number of features)

# Displaying the confusion matrix

matrix # Outputs the confusion matrix

# Importing the accuracy score function

from sklearn.metrics import accuracy\_score

# Calculating the accuracy of the model

acc = accuracy\_score(y\_test, y\_pred) # Computes the ratio of correctly predicted instances

print(acc) # Displays the accuracy score

# Importing the classification report function

from sklearn.metrics import classification\_report

# Generating the classification report

cr = classification\_report(y\_test, y\_pred) # Provides detailed performance metrics for each class

print(cr) # Displays the precision, recall, f1-score, and support for each class