Decision Tree

**Objective:**

The objective of this assignment is to apply Decision Tree Classification to a given dataset, analyse the performance of the model, and interpret the results.

**Tasks:**

1. Data Preparation:

Load the dataset into your preferred data analysis environment (e.g., Python with libraries like Pandas and NumPy).

**2. Exploratory Data Analysis (EDA):**

Perform exploratory data analysis to understand the structure of the dataset.

Check for missing values, outliers, and inconsistencies in the data.

Visualize the distribution of features, including histograms, box plots, and correlation matrices.

**3. Feature Engineering:**

If necessary, perform feature engineering techniques such as encoding categorical variables, scaling numerical features, or handling missing values.

**4. Decision Tree Classification:**

Split the dataset into training and testing sets (e.g., using an 80-20 split).

Implement a Decision Tree Classification model using a library like scikit-learn.

Train the model on the training set and evaluate its performance on the testing set using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score, ROC-AUC).

**5. Hyperparameter Tuning:**

Perform hyperparameter tuning to optimize the Decision Tree model. Experiment with different hyperparameters such as maximum depth, minimum samples split, and criterion.

**6. Model Evaluation and Analysis:**

Analyse the performance of the Decision Tree model using the evaluation metrics obtained.

Visualize the decision tree structure to understand the rules learned by the model and identify important features

**Interview Questions:**

1. What are some common hyperparameters of decision tree models, and how do they affect the model's performance?

Answer:

* **Max Depth (max\_depth)**:
  + Effect: Limits the depth of the tree. Controlling the depth of the tree helps to prevent overfitting (if the depth is too high) or underfitting (if the depth is too low).
* **Min Samples Split (min\_samples\_split)**:
  + Effect: The minimum number of samples required to split an internal node. Higher values prevent the model from learning overly specific patterns in the data, which helps to avoid overfitting.
* **Min Samples Leaf (min\_samples\_leaf)**:
  + Effect: The minimum number of samples required to be at a leaf node. Setting this parameter helps to smooth the model, especially for regression tasks, by ensuring that leaf nodes have enough data points.
* **Max Features (max\_features)**:
  + Effect: The number of features to consider when looking for the best split. Reducing the number of features can lead to a reduction in overfitting and a decrease in model variance.
* **Max Leaf Nodes (max\_leaf\_nodes)**:
  + Effect: Restricts the number of leaf nodes in the tree. Limiting leaf nodes can simplify the model and control overfitting.
* **Criterion (criterion)**:
  + Effect: The function used to measure the quality of a split. Common criteria are "gini" for the Gini impurity and "entropy" for information gain. The choice of criterion can affect the shape and performance of the tree.

2. What is the difference between the Label encoding and One-hot encoding?

Answer:

* **Label Encoding**:
  + Description: Converts categorical data into integer codes. Each unique category is assigned an integer value.
  + Usage: Suitable for ordinal data where the order matters. It can be used when the categorical variable has only two unique values (binary encoding).
  + Example: If you have categories ['cat', 'dog', 'mouse'], label encoding might convert these to [0, 1, 2].
  + Disadvantage: Can introduce a misleading ordinal relationship between categories that are actually nominal.
* **One-hot Encoding**:
  + Description: Converts categorical data into binary vectors. Each category is represented by a vector where one element is "hot" (1) and the others are "cold" (0).
  + Usage: Suitable for nominal data where no ordinal relationship exists. Used when the categorical variable has more than two unique values.
  + Example: If you have categories ['cat', 'dog', 'mouse'], one-hot encoding might convert these to:
    - cat: [1, 0, 0]
    - dog: [0, 1, 0]
    - mouse: [0, 0, 1]
  + Disadvantage: Can lead to a high-dimensional feature space when there are many unique categories, which might be computationally expensive and can cause the "curse of dimensionality".