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?

**Ans)** Decision tree models have several hyperparameters that can be tuned to improve performance and control the complexity of the tree. Some common hyperparameters include:

* **max\_depth:** This hyperparameter controls the maximum depth of the decision tree. A deeper tree can capture more complex relationships in the data but may also lead to overfitting, especially with noisy data. Setting a lower max\_depth can help prevent overfitting.
* **min\_samples\_split**: This hyperparameter sets the minimum number of samples required to split an internal node. Increasing this parameter can help prevent overfitting by requiring a certain number of samples in each node before a split is attempted.
* **min\_samples\_leaf**: This hyperparameter sets the minimum number of samples required to be at a leaf node. Similar to min\_samples\_split, increasing this parameter can help prevent overfitting by controlling the size of the leaf nodes.
* **max\_features**: This hyperparameter controls the number of features to consider when looking for the best split. Setting max\_features to a lower value can help reduce the variance of the model and prevent overfitting, especially in high-dimensional datasets.
* **criterion:** This hyperparameter determines the function used to measure the quality of a split. Common options include "gini" for the Gini impurity and "entropy" for information gain. Both criteria aim to maximize the purity of the resulting child nodes, but they may perform differently depending on the dataset.
* **splitter**: This hyperparameter specifies the strategy used to choose the split at each node. The two options are "best," which chooses the best split, and "random," which chooses the best random split. Choosing "random" can help prevent overfitting by introducing randomness into the tree-building process.

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

**Ans)** Label encoding and one-hot encoding are both techniques used to convert categorical variables into numerical format, but they differ in their approach and the way they handle categorical data.

**Label Encoding:**

* Label encoding assigns a unique integer to each category in a categorical variable.
* It is suitable for ordinal categorical variables where the categories have a meaningful order.
* The assigned integers are typically sequential starting from 0 or 1.
* It preserves the ordinality of the categories, meaning it assumes an inherent order among the categories.
* **Example**: If we have a categorical variable "Size" with categories ["Small", "Medium", "Large"], label encoding might map them to [0, 1, 2].

**One-Hot Encoding:**

* One-hot encoding creates binary dummy variables for each category in a categorical variable.
* It is suitable for nominal categorical variables where the categories do not have any meaningful order.
* For each category, one binary variable is created, where a value of 1 indicates the presence of the category, and 0 indicates absence.
* It does not assume any ordinal relationship among the categories and treats them as independent.
* **Example**: If we have a categorical variable "Color" with categories ["Red", "Green", "Blue"], one-hot encoding might create three binary variables: ["IsRed", "IsGreen", "IsBlue"].