Decision Tree | Assignment

Question 1: What is a Decision Tree, and how does it work in the context of classification?
ANSWER : A Decision Tree is a supervised learning algorithm used for classification and regression. It works by splitting data into branches based on feature values, forming a tree-like structure. Each leaf node represents a class label, and the model predicts by following decision rules from root to leaf.

Question 2: Explain the concepts of Gini Impurity and Entropy as impurity measures. How do they impact the splits in a Decision Tree?
ANSWER : Gini Impurity and Entropy are measures used to evaluate how mixed classes are in a node. Gini measures the probability of misclassification, while Entropy measures uncertainty or randomness. Lower values indicate purer nodes. These measures help the algorithm choose the best split that improves classification.

Question 3: What is the difference between Pre-Pruning and Post-Pruning in Decision Trees? Give one practical advantage of using each.
ANSWER : Pre-pruning stops tree growth early using limits like max depth to prevent overfitting. Post-pruning removes unnecessary branches after building a full tree. Pre-pruning reduces complexity early, while post-pruning improves generalization by trimming weak splits.

Question 4: What is Information Gain in Decision Trees, and why is it important for choosing the best split?
ANSWER : Information Gain is the reduction in entropy after a dataset is split on a feature. It measures how well a feature separates the classes. The split with the highest Information Gain is chosen because it produces more organized and accurate predictions.

Question 5: What are some common real-world applications of Decision Trees, and what are their main advantages and limitations?
ANSWER : Decision Trees are used in healthcare diagnosis, fraud detection, marketing prediction, and credit scoring. They are easy to understand, require little preprocessing, and handle different data types. However, they can overfit and are sensitive to small data changes.

Question 6: Write a Python program to:
● Load the Iris Dataset
● Train a Decision Tree Classifier using the Gini criterion
● Print the model’s accuracy and feature importances
Dataset Info:
● Iris Dataset for classification tasks (sklearn.datasets.load_iris() or provided CSV).
● Boston Housing Dataset for regression tasks (sklearn.datasets.load_boston() or provided CSV).
ANSWER :

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = DecisionTreeClassifier(criterion="gini")
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Feature Importances:", model.feature_importances_)
```

OUTPUT :-

```
Accuracy: 0.9
Feature Importances: [0.01668057 0.00771476 0.54039807 0.4352066 ]
```

Question 7: Write a Python program to:
● Load the Iris Dataset
● Train a Decision Tree Classifier with max_depth=3 and compare its accuracy to a fully-grown tree.

ANSWER :

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

full_tree = DecisionTreeClassifier()
full_tree.fit(X_train, y_train)
full_acc = accuracy_score(y_test, full_tree.predict(X_test))

limited_tree = DecisionTreeClassifier(max_depth=3)
limited_tree.fit(X_train, y_train)
limited_acc = accuracy_score(y_test, limited_tree.predict(X_test))

print("Full Tree Accuracy:", full_acc)
print("Depth=3 Tree Accuracy:", limited_acc)
```

OUTPUT :-

```
Full Tree Accuracy: 0.9333333333333333
Depth=3 Tree Accuracy: 0.9666666666666667
```

Question 8: Write a Python program to:
● Load the Boston Housing Dataset
● Train a Decision Tree Regressor
● Print the Mean Squared Error (MSE) and feature importances
ANSWER :

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

housing = fetch_california_housing()
X, y = housing.data, housing.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = DecisionTreeRegressor()
model.fit(X_train, y_train)

pred = model.predict(X_test)

print("MSE:", mean_squared_error(y_test, pred))
print("Feature Importances:", model.feature_importances_)
```

OUTPUT :-

```
MSE: 0.5320796499797722
Feature Importances: [0.51760159 0.05794115 0.04842583 0.02840019 0.0326416
0.13627441
 0.09347769 0.08523755]
```

Question 9: Write a Python program to:
● Load the Iris Dataset
● Tune the Decision Tree’s max_depth and min_samples_split using GridSearchCV
● Print the best parameters and the resulting model accuracy
ANSWER :

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

params = {
    "max_depth": [2, 3, 4, 5],
    "min_samples_split": [2, 5, 10]
}

grid = GridSearchCV(DecisionTreeClassifier(), params, cv=5)
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Best Accuracy:", grid.score(X_test, y_test))
```

OUTPUT :-

```
Best Parameters: {'max_depth': 3, 'min_samples_split': 2}
Best Accuracy: 0.9333333333333333
```

Question 10: Imagine you're working as a data scientist for a healthcare company that wants to predict whether a patient has a certain disease. You have a large dataset with mixed data types and some missing values. Explain the step-by-step process you would follow to:

● Handle the missing values
● Encode the categorical features
● Train a Decision Tree model
● Tune its hyperparameters
● Evaluate its performance And describe what business value this model could provide in the real-world setting.

ANSWER :

Step 1 — Handle missing values
Use imputation methods. For numerical features, use mean or median imputation. For categorical features, use the most frequent value. Advanced methods like KNN imputation can also be used for better accuracy.

Step 2 — Encode categorical features
Convert categorical variables using one-hot encoding or label encoding. One-hot encoding is safer when categories have no natural order.

Step 3 — Train Decision Tree
Split the dataset into training and testing sets. Train a Decision Tree classifier using cleaned and encoded data.

Step 4 — Hyperparameter tuning
Use GridSearchCV or RandomizedSearchCV to tune parameters such as max_depth, min_samples_split, and min_samples_leaf to avoid overfitting.

Step 5 — Evaluate performance
Measure accuracy, precision, recall, F1-score, and confusion matrix. In healthcare, recall is important to avoid missing disease cases.

Business value:
The model helps doctors identify high-risk patients early, reduces diagnostic errors,

improves treatment planning, and saves operational costs. It supports data-driven medical decisions and improves patient outcomes.