Assignment Code: DA-AG-012

Decision Tree Assignment

Q.1 What is a Decision Tree, and how does it work in the context of classification?

Ans.: A Decision Tree is a supervised learning algorithm used for classification & regression tasks. It models decisions as a tree-like structure, where each internal node represents a decision based on a feature, each branch represents an outcome of that decision, and each leaf node represents a final prediction.

- **It works in the context of classification as follows:**
  - In classification, the algorithm recursively splits the dataset based on a chosen feature and threshold that best separates the classes. The process continues until a stopping criterion is met (e.g., max depth, minimum samples). The prediction is made by following the path from the root to a leaf node for a given input.

Q.2 Explain the concepts of Gini Impurity and Entropy as impurity measures. How do they impact the splits in a Decision Tree?

Ans.:

- **Gini Impurity:**
  - Measures the probability that a randomly chosen sample would be misclassified if it were randomly labeled according to the class distribution in a node. Formula:
  - Gini = 1 - Σ (p_i)^2
  - Lower Gini means purer nodes.
- **Entropy Impurity:**
  - Measures the amount of uncertainty (or disorder) in a dataset. Formula:
  - Entropy = - Σ [p_i * log2(p_i)]
  - Higher entropy means more disorder.
- Impact on splits: Both measures guide the selection of the feature & threshold that results in the largest reduction in impurity, ensuring better class separation at each split.

Q.3 What is the difference between Pre-Pruning and Post-Pruning in Decision Trees? Give one practical advantage of using each.

Ans.:

- **Pre-Pruning:**
  - Stops the tree growth early based on conditions like max_depth, min_samples_split, or min_impurity_decrease.
  - Advantage: Prevents overfitting early & reduces computation time.
- **Post-Pruning:**
  - Grows the tree fully, then removes branches that do not improve performance on a validation set.
  - Advantage: Allows building a complex tree first, then simplifying it for better generalization.

Q.4 What is Information Gain in Decision Trees, and why is it important for choosing the best split?

Ans.: An Information Gain (IG) measures the reduction in impurity after a dataset is split on a feature.

- **It is calculated as:**
  - Information Gain = Impurity_parent - Σ [(n_k / n) * Impurity_child_k]
- It is important because it quantifies how much a split improves class separation, helping the algorithm choose the most informative features at each step.

Q.5 What are some common real-world applications of Decision Trees, and what are their main advantages and limitations?

Ans.:

- **Applications:**
  - Medical diagnosis
  - Credit risk assessment
  - Fraud detection
  - Customer segmentation

- **Advantages:**
  - Easy to understand & interpret
  - Can handle numerical & categorical data
  - Requires little data preprocessing
- **Limitations:**
  - Prone to overfitting
  - Can be biased toward features with many categories
  - Sensitive to small changes in data

Q.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**

Ans.:

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

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42
)
clf = DecisionTreeClassifier(criterion='gini', random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Feature Importances:", clf.feature_importances_)
```

```
Accuracy: 1.0
Feature Importances: [0.         0.01667014 0.90614339 0.07718647]
```

Q.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.

Ans.:

```
clf_depth3 = DecisionTreeClassifier(max_depth=3, random_state=42)
clf_depth3.fit(X_train, y_train)
acc_depth3 = accuracy_score(y_test, clf_depth3.predict(X_test))
clf_full = DecisionTreeClassifier(random_state=42)
clf_full.fit(X_train, y_train)
acc_full = accuracy_score(y_test, clf_full.predict(X_test))

print("Accuracy (max_depth=3):", acc_depth3)
print("Accuracy (fully grown):", acc_full)
```

```
Accuracy (max_depth=3): 1.0
Accuracy (fully grown): 1.0
```

Q.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

Ans.:

```
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_train, X_test, y_train, y_test = train_test_split(
    housing.data, housing.target, test_size=0.2, random_state=42
)
reg = DecisionTreeRegressor(random_state=42)
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Feature Importances:", reg.feature_importances_)
```

```
Mean Squared Error: 0.495235205629094
Feature Importances: [0.52850909 0.05188354 0.05297497 0.02866046 0.03051568 0.13083768
 0.09371656 0.08290203]
```

Q.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**

Ans.:

```python
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

param_grid = {
    'max_depth': [2, 3, 4, 5],
    'min_samples_split': [2, 3, 4]
}
grid_search = GridSearchCV(
    DecisionTreeRegressor(random_state=42),
    param_grid,
    cv=5,
    scoring='neg_mean_squared_error'
)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best negative Mean Squared Error:", grid_search.best_score_)
print("Best Mean Squared Error:", -grid_search.best_score_)
```

```
Best Parameters: {'max_depth': 5, 'min_samples_split': 2}
Best negative Mean Squared Error: -0.5080051413043611
Best Mean Squared Error: 0.5080051413043611
```

Q.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.**

Ans.:

- The Step by step for healthcare dataset:
  1) **Handle Missing Values:**
     a) Numerical: Fill with median or mean.
     b) Categorical: Fill with mode or create a separate category like "Unknown."
  2) **Encode Categorical Features:**
     a) Use One-Hot Encoding or Label Encoding depending on the model's requirements.
  3) **Train Decision Tree Model:**
     a) Split the dataset into training & testing sets.
     b) Initialize a DecisionTreeClassifier with default or chosen hyperparameters.
  4) **Tune Hyperparameters:**
     a) Use GridSearchCV or RandomizedSearchCV for parameters like max_depth, min_samples_split, criterion.
  5) **Evaluate Performance:**
     a) Use metrics like accuracy, precision, recall, F1-score, and confusion matrix.

Business Value:

- Enables early disease prediction, improving patient outcomes.
- Assists doctors in diagnosis by providing interpretable decision rules.
- Can reduce healthcare costs by identifying high-risk patients early.