Assignment Code: DA-AG-012

Decision Tree | **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 100

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 both classification and regression tasks. It is structured like an inverted tree:

- **Root Node:** Represents the entire dataset, which is split into subsets based on the most significant feature.
- **Internal Nodes:** Represent decision points where data is split according to certain feature values.
- **Leaf Nodes:** Represent the final output (class label in classification or value in regression).

Working in classification:

1. **Feature Selection:** The algorithm selects the best feature to split the dataset using a metric such as Gini Impurity or Entropy.
2. **Splitting:** The dataset is divided into subsets where the chosen feature best separates the classes.
3. **Recursive Process:** Steps 1–2 are repeated on each subset until stopping conditions are met (like maximum depth or no further gain in purity).
4. **Prediction:** For a new instance, the feature values are checked against decision rules from the root to a leaf, and the corresponding class label is assigned.

Example:
If we classify animals as "Mammal" or "Not Mammal", the first split might be "Has Hair?" → Yes → Mammal, No → Not Mammal.

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

Answer:

1. Gini Impurity:
Measures the probability of misclassifying a randomly chosen sample from the dataset.

Gini=1−∑i=1npi2Gini = 1 - \sum_{i=1}^{n} p_i^2Gini=1−i=1∑npi2

Where pip_ipi is the proportion of samples of class iii in the node.

- **Range:** 0 (pure node) to 0.5 (maximum impurity in binary classification).
- **Preference:** Tends to select larger partitions with better class purity.

2. Entropy:
Measures the amount of uncertainty in the data.

Entropy=−∑i=1npilog2(pi)Entropy = -\sum_{i=1}^n p_i \log_2(p_i)Entropy=−i=1∑npilog2(pi)

- **Range:** 0 (pure node) to 1 (maximum uncertainty for binary classification).
- **Preference:** More sensitive to class distribution changes.

Impact on splits:
Both aim to maximize **Information Gain**, selecting features that result in purer subsets. Gini is computationally simpler, while Entropy is more information-theoretic.

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 by setting limits like `max_depth`, `min_samples_split`, or `min_samples_leaf`.
- **Advantage:** Prevents overfitting before it happens, reduces training time.

Post-Pruning:

- Grows a complete tree, then removes branches that don't improve performance significantly.
- **Advantage:** Starts with maximum detail, then simplifies, often giving better accuracy–complexity balance.

Key difference: Pre-pruning limits complexity from the start; post-pruning optimizes after full growth.

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

Answer:

Definition:
Information Gain (IG) measures the reduction in impurity after splitting a dataset on a given feature.

IG=Impurityparent−∑knknImpuritychildkIG = Impurity_{parent} - \sum_{k} \frac{n_k}{n} Impurity_{child_k}IG=Impurityparent−k∑nnkImpuritychildk

Importance:

- Guides feature selection at each split.
- Higher IG means a feature provides better separation of classes.
- Ensures the tree is built using the most informative features, improving classification accuracy.

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

Answer:

Applications:

- Medical diagnosis (predicting disease presence).
- Credit risk assessment in banking.
- Fraud detection in finance.
- Customer churn prediction in telecom.

Advantages:

- Easy to interpret and visualize.
- Handles both categorical and numerical features.
- Requires little preprocessing (no scaling needed).

Limitations:

- Prone to overfitting without pruning.
- Sensitive to small changes in data.
- Can be biased toward features with many categories.

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).

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

(*Include your Python code and output in the code box below.*)

Answer:

```
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

# Load dataset

data = load_iris()

X, y = data.data, data.target

# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y,

test_size=0.2, random_state=42)


# Model training

model = DecisionTreeClassifier(criterion='gini',
```

random_state=42)

model.fit(X_train, y_train)

Results

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Feature Importances:",

model.feature_importances_)

output

Accuracy: 1.0

Feature Importances: [0.02 0.02 0.44 0.52]

\

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.

(*Include your Python code and output in the code box below.*)

Answer:

```
# Limited depth tree
model_limited = DecisionTreeClassifier(max_depth=3, random_state=42)
model_limited.fit(X_train, y_train)

# Full tree
model_full = DecisionTreeClassifier(random_state=42)
model_full.fit(X_train, y_train)

print("Accuracy (max_depth=3):", accuracy_score(y_test, model_limited.predict(X_test)))
print("Accuracy (Full Tree):", accuracy_score(y_test, model_full.predict(X_test)))
```

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

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.datasets import load_boston
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Load data
data = load_boston()
X, y = data.data, data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(X_train, y_train)

# Predictions & MSE
y_pred = regressor.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("Feature Importances:", regressor.feature_importances_)
```

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

(*Include your Python code and output in the code box below.*)

Answer:

```
from sklearn.model_selection import GridSearchCV

params = {
    'max_depth': [2, 3, 4, 5, None],
    'min_samples_split': [2, 4, 6]
}
```

```
grid = GridSearchCV(DecisionTreeClassifier(random_state=42), params, cv=5)
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Best Accuracy:", grid.best_score_)
```

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:

1. Handle Missing Values:

- Numerical: Fill with mean/median.
- Categorical: Fill with mode or create an "Unknown" category.

2. Encode Categorical Features:

- Use One-Hot Encoding for nominal variables.
- Use Label Encoding for ordinal variables.

3. Train Decision Tree Model:

- Use `criterion='gini'` or `'entropy'` and tune `max_depth`.

4. Tune Hyperparameters:

- Use GridSearchCV to adjust `max_depth`, `min_samples_split`, `min_samples_leaf`.

5. Evaluate Performance:

- Accuracy, Precision, Recall, F1-score.
- ROC-AUC for binary classification. In healthcare, **Recall** is critical to reduce false negatives.

Business Value:

- Assists doctors in early detection.
- Reduces manual diagnostic workload.
- Supports targeted treatment, improving patient outcomes.