Decision Tree | Assignment

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

Ans:

A Decision Tree is a supervised machine learning algorithm used for both classification and regression problems. It represents decisions in the form of a tree-like structure, where internal nodes represent conditions on features, branches represent decision rules, and leaf nodes represent the final output or class label.

How a Decision Tree Works in Classification-

In a classification problem, a Decision Tree works by repeatedly splitting the dataset into smaller subsets based on feature values.

Step-by-step working:

- The algorithm starts at the root node containing the full dataset.
- It selects the best feature to split the data based on an impurity measure such as Gini Impurity or Entropy.
- The dataset is divided into subsets according to the selected feature.
- This process is repeated recursively for each subset.
- The splitting stops when a stopping condition is met (pure node, max depth reached, or minimum samples).
- Each leaf node assigns a class label based on the majority class of the samples in that node.

Key Characteristics:

- Easy to understand and interpret
- Mimics human decision-making
- Works with both numerical and categorical data

Question 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: Gini Impurity measures the probability that a randomly chosen sample would be incorrectly classified.

Formula: $Gini = 1 - \Sigma(p_i^2)$
Where p_i is the probability of class i.

Interpretation:

- Gini = 0 → Node is pure
- Lower Gini → Better split

Entropy: Entropy measures the level of disorder or randomness in the dataset.
Formula: Entropy = $-\Sigma(p_i \log_2 p_i)$

Interpretation:

- Entropy = 0 → Perfectly pure node
- Higher entropy → More mixed classes

Impact on Decision Tree Splits:

- Both metrics aim to reduce impurity
- The algorithm chooses the split that gives maximum impurity reduction
- Gini is computationally faster
- Entropy is more sensitive to class imbalance

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

Ans:

1. **Pre-Pruning (Early Stopping):** Pre-pruning stops the tree growth early by setting constraints.

Common techniques:

- Maximum depth
- Minimum samples per split
- Minimum impurity decrease

Practical Advantage:

- Reduces overfitting
- Faster training time

2. **Post-Pruning:** Post-pruning allows the tree to grow fully and then trims unnecessary branches.

Common techniques:

- Cost complexity pruning
- Reduced error pruning

Practical Advantage:

- Often produces better generalization
- More accurate than pre-pruning

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

Ans:

Information Gain measures the reduction in entropy after a dataset is split on a feature.

Formula: Information Gain = Entropy(parent) − Weighted Entropy(children)

Importance of Information Gain

- Helps identify the most informative feature
- Ensures optimal splits
- Improves classification accuracy
- Reduces randomness at each node

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

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).**

Ans:

Common Applications

- Healthcare: Diagnosing diseases based on patient symptoms.
- Finance: Credit scoring and fraud detection.
- Marketing: Customer churn prediction (identifying which customers might leave).

Advantages

- Easy to interpret
- Handles non-linear relationships
- Requires minimal data preprocessing

Limitations

- Prone to overfitting
- Unstable with small data changes
- Lower accuracy compared to ensemble models

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**

Ans:

```
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

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

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

# Train model
model = DecisionTreeClassifier(criterion='gini', random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

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

# Feature Importances
print("Feature Importances:", model.feature_importances_)
```

Output:

- Accuracy ≈ 1.0
- Displays importance of each feature

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.**

Ans:

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

# Fully grown model
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)))
```

Observation:

- Fully grown tree may overfit
- Limited depth provides better generalization

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**

Ans:

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

boston = load_boston()
X, y = boston.data, boston.target

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

model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("MSE:", mean_squared_error(y_test, y_pred))
print("Feature Importances:", model.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**

Ans:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
'max_depth': [2, 3, 4, 5, None],
'min_samples_split': [2, 5, 10]
}

grid = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, 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**

Ans:

1. Handling Missing Values

Missing values are common in healthcare datasets due to incomplete medical records or skipped tests.

Steps to Handle Missing Values

a) Identify Missing Data

- Check missing values using summary statistics or functions like isnull() in Python.
- Understand which features have a high percentage of missing values.

b) Numerical Features

- Replace missing values with:
  - Mean if data is normally distributed
  - Median if data contains outliers
- Example: Missing blood pressure or cholesterol values can be filled with the median.

c) Categorical Features

- Replace missing values with:
  - Mode (most frequent category)
  - A new category such as “Unknown”
- This ensures no data is lost.

Why This Step Is Important

- Prevents data loss
- Ensures the Decision Tree can process all records
- Improves model accuracy and stability

2. Encoding the Categorical Features

Machine learning models require numerical input, so categorical variables must be encoded.

Encoding Techniques Used

a) Label Encoding

- Used **for ordinal data (e.g., disease severity: mild < moderate < severe).**
- **Converts categories into integer values**.

b) One-Hot Encoding

- Used for nominal data (e.g., gender, blood group).
- Creates binary columns for each category.

Importance of Encoding

- Converts real-world medical information into machine-readable format
- Avoids introducing incorrect order in non-ordinal features

3. Training a Decision Tree Model

After preprocessing, the dataset is ready for model training.

Steps to Train the Model

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

- Choose a Decision Tree Classifier.
- Select an impurity measure:
    - Gini Impurity for faster computation
    - Entropy for more informative splits
- Train the model using the training dataset.

Why Decision Trees Are Suitable in Healthcare

- Easy to interpret by doctors and clinicians
- Decision rules can be explained clearly
- Handles non-linear relationships effectively

4. Hyperparameter Tuning

A fully grown Decision Tree may overfit the data, so tuning is essential.

Important Hyperparameters to Tune

- max_depth – Controls tree depth
- min_samples_split – Minimum samples required to split a node
- min_samples_leaf – Minimum samples required in a leaf node
- criterion – Gini or Entropy

Tuning Method

- Use GridSearchCV with cross-validation
- Select the parameter combination that gives the best performance

Benefits of Tuning

- Reduces overfitting
- Improves generalization to new patients
- Increases model reliability

5. Evaluating Model Performance

Evaluating the model ensures it performs well on unseen data.

Key Evaluation Metrics

- Accuracy – Overall correctness of predictions
- Precision – How many predicted disease cases are actually correct
- Recall (Sensitivity) – Ability to identify patients with the disease
- F1-score – Balance between precision and recall
- Confusion Matrix – Shows correct and incorrect predictions

Importance in Healthcare

- High recall is crucial to avoid missing actual disease cases
- Balanced evaluation prevents harmful misdiagnosis

6. Business Value in the Real-World Healthcare Setting

Key Business and Clinical Benefits

- **Early Disease Detection**
    - Helps doctors identify high-risk patients early
- **Improved Patient Outcomes**
    - Early treatment leads to better recovery rates
- **Cost Reduction**
    - Prevents expensive late-stage treatments
- **Decision Support for Doctors**
    - Provides explainable insights for clinical decisions
- **Scalability**
    - Can analyze thousands of patient records quickly