Assignment Code: DA-AG-012

Decision Tree | **Assignment**

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** and **regression** tasks. In classification, it helps predict which category a given data point belongs to by learning decision rules from the data features.

How It Works (Step-by-Step for Classification)

1. **Root Node**
   - Starts with the entire dataset.
   - Chooses the best feature to split on using impurity metrics like *Gini* or *Entropy*.
2. **Splitting Criteria**
   - At each node, the algorithm selects the feature and threshold that **maximizes class separation** (e.g., highest Information Gain or Gini reduction).
3. **Branching**
   - Data is partitioned into subsets.
   - Each subset forms a new branch with more specific conditions.
4. **Leaf Node**
   - Splitting stops when:
     - All samples belong to the same class.
     - Impurity is minimal.
     - Predefined constraints are reached (e.g., max depth).
   - Final prediction is made based on majority class in that leaf.

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 from the node would be incorrectly classified if labeled randomly according to the distribution of labels in that node.

Formula: $\text{Gini} = 1 - \sum_{i=1}(p_i)2$

- (p_i) is the probability of class (i) in the node

Example: A node has:

- 60% Class A
- 40% Class B

Gini = (1 - (0.6$^{2\,+}$0.4^2) = 0.48)

Entropy comes from information theory. It measures the amount of uncertainty or disorder at a node.

Formula: $Entropy = -\sum_{i=1}^{N} p_i \cdot \log_2(p_i)$

Example: Same distribution:

$$Entropy = -(0.6 \cdot \log_2(0.6) + 0.4 \cdot \log_2(0.4)) \approx 0.97$$

Impact on Tree Splits

Both metrics are used to evaluate which feature and threshold split the node most effectively:

- **Lower impurity after the split** → better feature
- Algorithms try all possible splits, compute resulting impurity, and select the one that **maximizes purity** in child nodes.

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

Ans: **Pre-Pruning vs Post-Pruning**

Feature	**Pre-Pruning**	**Post-Pruning**
Timing	Happens *while* building the tree	Happens *after* the tree is fully grown
Method	Stops splitting if conditions fail	Removes subtrees based on validation checks
Criteria Used	Max depth, min samples, Gini/Entropy gain	Validation accuracy, complexity trade-off
Outcome	May prevent overfitting early	Cleans up a complex model post hoc

Practical Advantages

- **Pre-Pruning Advantage**
  Speeds up training time — By halting unnecessary splits early, it reduces computational cost, especially useful for large datasets or real-time systems.
- **Post-Pruning Advantage**
  Improves generalization — By trimming branches that don't boost validation accuracy, it sharpens the model's ability to perform well on unseen data.

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

Ans: **Information Gain (IG)** measures the reduction in entropy when a dataset is split on a particular feature.

Entropy quantifies uncertainty — so IG tells us how *much that uncertainty drops* when we make a split.

Formula:

Information Gain = Entropy(parent) - ∑ (Weighted Entropy(children))

We compute the total entropy before the split, then subtract the weighted sum of the entropies after splitting the data.

Why Is It Important?

Because it acts like a magnet for the **best feature to split on**.

- If IG is high → the split creates purer child nodes → great candidate for a split!
- If IG is low → the feature doesn't reduce uncertainty much → not ideal.

In short: **Higher Information Gain = More confident and meaningful split**.

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

Ans: **Decision Trees** are a type of supervised learning algorithm commonly used for both classification and regression problems across various industries.

Some common applications include:

- **Healthcare**: Used in diagnostics, predicting patient outcomes, and genomics.
- **Finance**: Applied in risk assessment, fraud detection, and option pricing.
- **Marketing**: Utilized for predicting customer churn and customer segmentation.
- **E-commerce**: Employed in recommendation systems.
- **Manufacturing**: Used for quality control.
- **Education**: Applied in predicting student performance.

Advantages of Decision Trees:

- Easy to Understand and Interpret: Visually intuitive and accessible.
- Handle Both Numerical and Categorical Data: Versatile for different data types.
- Require Little Data Preparation: Don't need extensive pre-processing.
- Handle Non-linear Relationships: Can capture complex relationships.

- Good for Feature Importance: Rank features by importance.
- Robust to Outliers: Relatively unaffected by outliers.
- Can Handle Missing Values: Can work with datasets that have missing data.

Limitations of Decision Trees

- Overfitting: Can capture noise, leading to poor performance on new data. Pruning or limiting depth helps.
- Instability: Small data changes can significantly alter the tree.
- Bias towards features with many categories: May favor features with numerous values.
- Difficulty in capturing complex interactions: May struggle with intricate feature interactions compared to other algorithms.
- Computationally Expensive for Large Datasets: Building large trees can be intensive.

6. - 9.

GitHub Link

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: **Handle Missing Values**

- **Identify missing values** using df.isnull().sum() for a clear picture.

- **Numerical features:**
    - Use **mean/median imputation** for continuous data.
    - For skewed distributions, prefer median to avoid distortion.
- **Categorical features:**
    - Replace missing entries with the **mode** or a distinct label like "Missing".
- Consider using **SimpleImputer** from sklearn.impute for streamlined preprocessing.

Encode Categorical Features

- **Label Encoding**: For ordinal features with a natural order.
- **One-Hot Encoding**: For nominal features (non-ordered), using pd.get_dummies() or OneHotEncoder.

Train a Decision Tree Model

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=2)

model.fit(X_train, y_train)
```

- No need to scale features, which is a bonus with tree-based models.
- Decision trees naturally handle mixed data types after encoding.

Tune Hyperparameters

Use **GridSearchCV** or **RandomizedSearchCV** to optimize parameters like:

- max_depth: Controls overfitting.
- min_samples_split and min_samples_leaf: Improve generalization.
- criterion: Try both 'gini' and 'entropy'.

- class_weight: Important if your dataset is imbalanced.

```
from sklearn.model_selection import GridSearchCV


param_grid = {
    'max_depth': [3, 5, None],
    'min_samples_split': [2, 5, 10],
    'criterion': ['gini', 'entropy']
}
grid_search = GridSearchCV(DecisionTreeClassifier(random_state=2), param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

Evaluate Performance

Use a mix of metrics:

- **Accuracy**: Good for balanced datasets.
- **Precision/Recall/F1 Score**: Crucial for disease prediction where false negatives matter.
- **Confusion Matrix** and **ROC AUC**: For visual insights.

```
from sklearn.metrics import classification_report, roc_auc_score


print(classification_report(y_test, best_model.predict(X_test)))
print("ROC AUC:", roc_auc_score(y_test, best_model.predict_proba(X_test)[:,1]))
```

Business Value in Healthcare

A well-tuned model like this could:

- **Enable early detection** and intervention for at-risk patients.
- Support **personalized treatment plans**.
- Help hospitals **prioritize resources** and manage patient loads.
- Provide **predictive insights** for chronic condition management.
- Reduce operational costs by **automating screening** and triage.

By making disease prediction more proactive, the company could increase patient survival rates and reduce long-term care expenses.