UTKARSH CHAUBE 20214195 CSED

ASSIGNMENT 6

1. Compute the split point for each attribute in the dataset using the following strategies:

   a. Information Gain

   b. Gini Indexes

   c. Gain Ratio

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
# Load the dataset
df = pd.read_csv("vehicle.csv")


# Preprocessing (if required)
# Check for missing values
print(df.isnull().sum())


# Handle missing values (if any)
# For simplicity, let's drop rows with missing values
df.dropna(inplace=True)


# Separate features (X) and target variable (y)
X = df.drop(columns=['class'])  # Assuming 'class' is the target variable
y = df['class']


# Initialize the decision tree classifier
clf = DecisionTreeClassifier()


# Fit the classifier to the data
clf.fit(X, y)
```

```
# Function to compute Information Gain
def information_gain(y, y_splits):
    entropy_parent = entropy(y)
    total_instances = len(y)
    weighted_entropy_children = sum((len(y_split) / total_instances) *
entropy(y_split) for y_split in y_splits)
    return entropy_parent - weighted_entropy_children
# Function to compute Gini Index
def gini_index(y, y_splits):
    gini_parent = gini_impurity(y)
    total_instances = len(y)
    weighted_gini_children = sum((len(y_split) / total_instances) *
gini_impurity(y_split) for y_split in y_splits)
    return gini_parent - weighted_gini_children


# Function to compute entropy
def entropy(y):
    classes, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return -sum(p * np.log2(p) for p in probabilities)


# Function to compute Gini impurity
def gini_impurity(y):
    classes, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return 1 - sum(p**2 for p in probabilities)
# Function to compute split points using Information Gain, Gini Index, and
Gain Ratio
def compute_split_points(X, y):
    split_points = {}
    for col in X.columns:
        # Assuming each attribute is numeric
        values = X[col].unique()
        for value in values:
            # Split the dataset based on the attribute value
            left_indices = X[col] < value
```

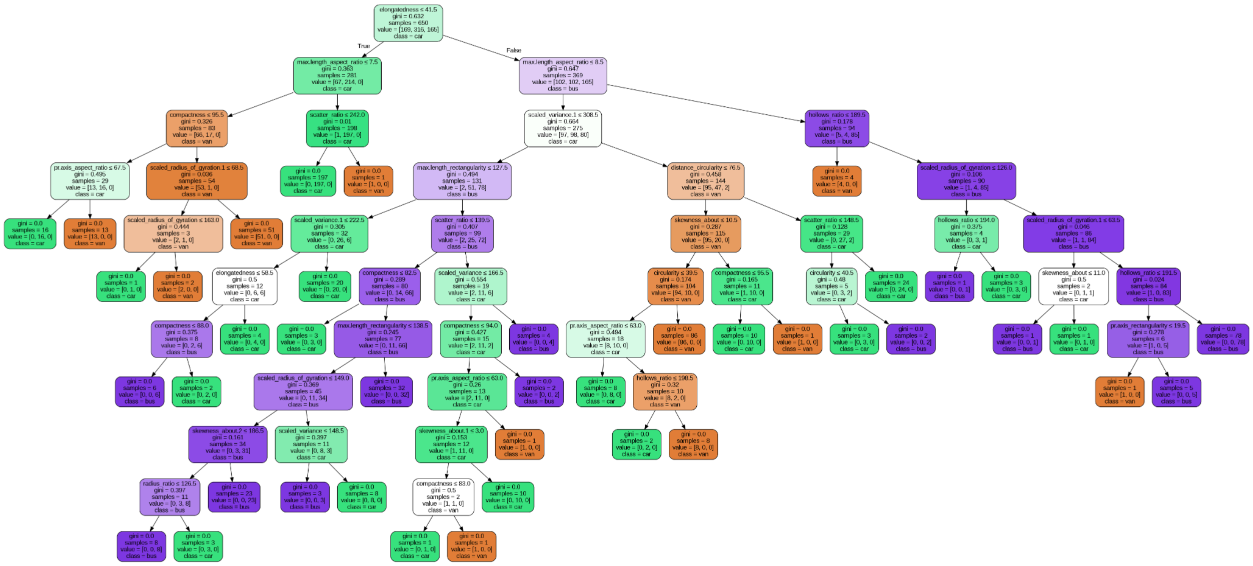
```
           right_indices = ~left_indices
           y_splits = [y[left_indices], y[right_indices]]
           # Compute metrics for split points
           info_gain = information_gain(y, y_splits)
           gini_idx = gini_index(y, y_splits)
           gain_ratio = info_gain / (entropy(X[col]) + 1e-10)  # Add small
value to avoid division by zero

           split_points[(col, value)] = {'Information Gain': info_gain,
'Gini Index': gini_idx, 'Gain Ratio': gain_ratio}

   return split_points


# Compute split points for each attribute
split_points = compute_split_points(X, y)


# Print split points
for key, value in split_points.items():
   print(f"Split Point for {key}: {value}")
```

2. Design module for creating the decision tree and its representation in graphical format for the following cases:

   a. Binary Tree (each node split into exactly two branches).
   b. General Tree (each node may split into more than two branches depending on count nominal labels corresponding attributes).

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import export_graphviz
import graphviz


# Load the dataset
df = pd.read_csv("vehicle.csv")
```

```python
# Check for missing values before preprocessing
print("Missing values before preprocessing:")
print(df.isnull().sum())


# Drop rows with missing values
df.dropna(inplace=True)


# Check for missing values after preprocessing
print("\nMissing values after preprocessing:")
print(df.isnull().sum())


# Split the dataset into features and target variable
X = df.drop(columns=['class'])
y = df['class']


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Initialize the decision tree classifier
clf = DecisionTreeClassifier()


# Fit the classifier to the training data
clf.fit(X_train, y_train)


# Predict the labels of the test set
y_pred = clf.predict(X_test)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))


# Visualize the decision tree
dot_data = export_graphviz(clf, out_file=None,
                           feature_names=X.columns,
                           class_names=y.unique(),
                           filled=True, rounded=True,
                           special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("vehicle_decision_tree", format='png', cleanup=True)
```


elongatedness ≤ 41.5 gini = 0.632 samples = 650 value = [169, 316, 165] class = car
True
False
max.length_aspect_ratio ≤ 7.5 gini = 0.363 samples = 281 value = [67, 214, 0] class = car
max.length_aspect_ratio ≤ 8.5 gini = 0.647 samples = 369 value = [102, 102, 165] class = bus
compactness ≤ 95.5 gini = 0.326 samples = 83 value = [66, 17, 0] class = van
scatter_ratio ≤ 242.0 gini = 0.01 samples = 198 value = [1, 197, 0] class = car
scaled_variance.1 ≤ 308.5 gini = 0.664 samples = 275 value = [97, 98, 80] class = car
hollows_ratio ≤ 189.5 gini = 0.178 samples = 94 value = [5, 4, 85] class = bus
pr.axis_aspect_ratio ≤ 67.5 gini = 0.495 samples = 29 value = [13, 16, 0] class = car
scaled_radius_of_gyration.1 ≤ 68.5 gini = 0.036 samples = 54 value = [53, 1, 0] class = van
gini = 0.0 samples = 197 value = [0, 197, 0] class = car
gini = 0.0 samples = 1 value = [1, 0, 0] class = van
max.length_rectangularity ≤ 127.5 gini = 0.494 samples = 131 value = [2, 51, 78] class = bus
distance_circularity ≤ 76.5 gini = 0.458 samples = 144 value = [95, 47, 2] class = van
gini = 0.0 samples = 4 value = [4, 0, 0] class = van
scaled_radius_of_gyration ≤ 126.0 gini = 0.106 samples = 90 value = [1, 4, 85] class = bus
gini = 0.0 samples = 16 value = [0, 16, 0] class = car
gini = 0.0 samples = 13 value = [13, 0, 0] class = van
scaled_radius_of_gyration ≤ 163.0 gini = 0.444 samples = 3 value = [2, 1, 0] class = van
gini = 0.0 samples = 51 value = [51, 0, 0] class = van
scaled_variance.1 ≤ 222.5 gini = 0.305 samples = 32 value = [0, 26, 6] class = car
scatter_ratio ≤ 139.5 gini = 0.407 samples = 99 value = [2, 25, 72] class = bus
skewness_about ≤ 10.5 gini = 0.287 samples = 115 value = [95, 20, 0] class = van
scatter_ratio ≤ 148.5 gini = 0.128 samples = 29 value = [0, 27, 2] class = car
hollows_ratio ≤ 194.0 gini = 0.375 samples = 4 value = [0, 3, 1] class = car
scaled_radius_of_gyration.1 ≤ 63.5 gini = 0.046 samples = 86 value = [1, 1, 84] class = bus
gini = 0.0 samples = 1 value = [0, 1, 0] class = car
gini = 0.0 samples = 2 value = [2, 0, 0] class = van
elongatedness ≤ 58.5 gini = 0.5 samples = 12 value = [0, 6, 6] class = car
gini = 0.0 samples = 20 value = [0, 20, 0] class = car
compactness ≤ 82.5 gini = 0.289 samples = 80 value = [0, 14, 66] class = bus
scaled_variance ≤ 166.5 gini = 0.554 samples = 19 value = [2, 11, 6] class = car
circularity ≤ 39.5 gini = 0.174 samples = 104 value = [94, 10, 0] class = van
compactness ≤ 95.5 gini = 0.165 samples = 11 value = [1, 10, 0] class = car
circularity ≤ 40.5 gini = 0.48 samples = 5 value = [0, 3, 2] class = car
gini = 0.0 samples = 24 value = [0, 24, 0] class = car
gini = 0.0 samples = 1 value = [0, 0, 1] class = bus
gini = 0.0 samples = 3 value = [0, 3, 0] class = car
skewness_about ≤ 11.0 gini = 0.5 samples = 2 value = [0, 1, 1] class = car
hollows_ratio ≤ 191.5 gini = 0.024 samples = 84 value = [1, 0, 83] class = bus
compactness ≤ 88.0 gini = 0.375 samples = 8 value = [0, 2, 6] class = bus
gini = 0.0 samples = 4 value = [0, 4, 0] class = car
gini = 0.0 samples = 3 value = [0, 3, 0] class = car
max.length_rectangularity ≤ 138.5 gini = 0.245 samples = 77 value = [0, 11, 66] class = bus
compactness ≤ 94.0 gini = 0.427 samples = 15 value = [2, 11, 2] class = car
gini = 0.0 samples = 4 value = [0, 0, 4] class = bus
pr.axis_aspect_ratio ≤ 63.0 gini = 0.494 samples = 18 value = [8, 10, 0] class = car
gini = 0.0 samples = 86 value = [86, 0, 0] class = van
gini = 0.0 samples = 10 value = [0, 10, 0] class = car
gini = 0.0 samples = 1 value = [1, 0, 0] class = van
gini = 0.0 samples = 3 value = [0, 3, 0] class = car
gini = 0.0 samples = 2 value = [0, 0, 2] class = bus
gini = 0.0 samples = 1 value = [0, 0, 1] class = bus
gini = 0.0 samples = 1 value = [0, 1, 0] class = car
pr.axis_rectangularity ≤ 19.5 gini = 0.278 samples = 6 value = [1, 0, 5] class = bus
gini = 0.0 samples = 78 value = [0, 0, 78] class = bus
gini = 0.0 samples = 6 value = [0, 0, 6] class = bus
gini = 0.0 samples = 2 value = [0, 2, 0] class = car
scaled_radius_of_gyration ≤ 149.0 gini = 0.369 samples = 45 value = [0, 11, 34] class = bus
gini = 0.0 samples = 32 value = [0, 0, 32] class = bus
pr.axis_aspect_ratio ≤ 63.0 gini = 0.26 samples = 13 value = [2, 11, 0] class = car
gini = 0.0 samples = 2 value = [0, 0, 2] class = bus
gini = 0.0 samples = 8 value = [0, 8, 0] class = car
hollows_ratio ≤ 198.5 gini = 0.32 samples = 10 value = [8, 2, 0] class = van
gini = 0.0 samples = 1 value = [1, 0, 0] class = van
gini = 0.0 samples = 5 value = [0, 0, 5] class = bus
skewness_about.2 ≤ 186.5 gini = 0.161 samples = 34 value = [0, 3, 31] class = bus
scaled_variance ≤ 148.5 gini = 0.397 samples = 11 value = [0, 8, 3] class = car
skewness_about.1 ≤ 3.0 gini = 0.153 samples = 12 value = [1, 11, 0] class = car
gini = 0.0 samples = 1 value = [1, 0, 0] class = van
gini = 0.0 samples = 2 value = [0, 2, 0] class = car
gini = 0.0 samples = 8 value = [8, 0, 0] class = van
radius_ratio ≤ 126.5 gini = 0.397 samples = 11 value = [0, 3, 8] class = bus
gini = 0.0 samples = 23 value = [0, 0, 23] class = bus
gini = 0.0 samples = 3 value = [0, 0, 3] class = bus
gini = 0.0 samples = 8 value = [0, 8, 0] class = car
compactness ≤ 83.0 gini = 0.5 samples = 2 value = [1, 1, 0] class = van
gini = 0.0 samples = 10 value = [0, 10, 0] class = car
gini = 0.0 samples = 8 value = [0, 0, 8] class = bus
gini = 0.0 samples = 3 value = [0, 3, 0] class = car
gini = 0.0 samples = 1 value = [0, 1, 0] class = car
gini = 0.0 samples = 1 value = [1, 0, 0] class = van

3. Design module which predicts the class label of unknown and unseen data using tree traversal or any other techniques.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
# Load the dataset
df = pd.read_csv('vehicle.csv')
# Drop rows with missing values
df.dropna(inplace=True)
# Split dataset into features and target variable
X = df.drop(columns=['class'])
y = df['class']


# Split data into training and testing subsets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Create a decision tree classifier
clf = DecisionTreeClassifier()


# Train the classifier on the training data
clf.fit(X_train, y_train)


# Predict the classes of testing data
y_pred = clf.predict(X_test)


# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```