Assignment No. 3

Problem Statement: Implement and analyze the Decision Tree algorithm for classification and regression.

Objective:
Understand the working of Decision Trees.
Implement Decision Tree models for both classification and regression tasks.
Evaluate model performance and analyze how different parameters affect accuracy.

Prerequisite :

Python environment with libraries such as numpy, pandas, matplotlib, seaborn, and sklearn. Basic understanding of machine learning and Decision Trees.

Theory :

A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It is a tree-like model where data is split based on feature values, leading to a final decision at leaf nodes.

How Decision Trees Work?

1. Feature Selection: The algorithm selects the best feature to split the data at each node using criteria like:
    - Entropy & Information Gain (for classification)
    - Gini Impurity (for classification)
    - Mean Squared Error (MSE) (for regression)
2. Recursive Splitting: The dataset is split into smaller subsets based on the selected feature until it meets a stopping condition (e.g., max depth, min samples per leaf).
3. Leaf Nodes: Once the splitting process stops, the leaf nodes represent the final classification or predicted value.

Key Concepts

1. Information Gain & Entropy

- Entropy measures the impurity of a dataset. Lower entropy means purer data.
- Information Gain (IG) is used to determine the best feature to split the data. The feature with the highest IG is selected.

  2. Gini Impurity

- Measures how often a randomly chosen element would be incorrectly classified.
- It ranges from 0 (pure dataset) to 1 (impure dataset).

3. Pruning

Pruning helps prevent overfitting by reducing the size of the tree. There are two main types:

- Pre-Pruning (Early Stopping): Stops growing the tree if conditions like minimum samples per node are met.
- Post-Pruning (Prune After Training): Removes less important branches after the tree is built.

4. Regression using Decision Trees

For regression, Decision Trees predict continuous values instead of classes. The Mean Squared Error (MSE) is used as a splitting criterion:

Advantages of Decision Trees

Simple to understand and interpret.
Can handle both numerical and categorical data.
Requires minimal data preprocessing (no need for feature scaling).
Can model non-linear relationships.

Disadvantages of Decision Trees

Prone to overfitting, especially with deep trees.
Sensitive to small variations in data.
Decision boundaries may not be smooth compared to other models like SVM.

CODE & OUTPUT :

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

[3]: file_path = "/Users/pranavashokdivekar/this_mac/Machine Learning/car_evaluation.csv"
     df = pd.read_csv(file_path)

[4]: df.shape

[4]: (1727, 7)
```

[8]:

```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


for col in col_names:

    print(df[col].value_counts())
```

```
buying
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
maint
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
doors
3        432
4        432
5more    432
2        431
Name: count, dtype: int64
persons
4       576
more    576
2       575
Name: count, dtype: int64
lug_boot
med      576
big      576
small    575
Name: count, dtype: int64
safety
med     576
high    576
low     575
Name: count, dtype: int64
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

[9]:

```
df['class'].value_counts()
```

[9]:

	count
class	
unacc	1209
acc	384
good	69
vgood	65

dtype: int64

[10]:

```
# check missing values in variables

df.isnull().sum()
```

[10]:

	0
buying	0
maint	0
doors	0
persons	0
lug_boot	0
safety	0
class	0

dtype: int64

[11]:
```
X = df.drop(['class'], axis=1)

y = df['class']
```

[12]:
```
# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

[13]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

[13]: ((1157, 6), (570, 6))

[14]:
```
# check data types in X_train

X_train.dtypes
```

[14]:

	0
buying	object
maint	object
doors	object
persons	object
lug_boot	object
safety	object

dtype: object

[15]:
```
X_train.head()
```

[15]:

	buying	maint	doors	persons	lug_boot	safety
83	vhigh	vhigh	5more	2	med	low
48	vhigh	vhigh	3	more	med	med
468	high	vhigh	3	4	small	med
155	vhigh	high	3	more	med	low
1043	med	high	4	more	small	low

[19]:
```
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'])


X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

[20]:
```
X_train.head()
```

[20]:

	buying	maint	doors	persons	lug_boot	safety
83	1	1	1	1	1	1
48	1	1	2	2	1	2
468	2	1	2	3	2	2
155	1	2	2	2	1	1
1043	3	2	3	2	2	1

[21]:
```
X_test.head()
```

[21]:

	buying	maint	doors	persons	lug_boot	safety
599	2	2	3	1	3	1
932	3	1	3	3	3	1
628	2	2	1	1	3	3
1497	4	2	1	3	1	2
1262	3	4	3	2	1	1

```
[22]: from sklearn.tree import DecisionTreeClassifier

[23]: clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


      # fit the model
      clf_gini.fit(X_train, y_train)

[23]: DecisionTreeClassifier(max_depth=3, random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

DecisionTreeClassifier
?Documentation for DecisionTreeClassifieriFitted

```
DecisionTreeClassifier(max_depth=3, random_state=0)

[24]: y_pred_gini = clf_gini.predict(X_test)

[25]: from sklearn.metrics import accuracy_score

      print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred_gini)))

      Model accuracy score with criterion gini index: 0.8053

[26]: y_pred_train_gini = clf_gini.predict(X_train)

      y_pred_train_gini

[26]: array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
            dtype=object)

[27]: print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))

      Training-set accuracy score: 0.7848

[28]: # print the scores on training and test set

      print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))

      print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))

      Training set score: 0.7848
      Test set score: 0.8053
```

```
[33]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test, y_pred_gini)

      print("Confusion Matrix:")
      print(cm)

      Confusion Matrix:
      [[ 71   0  56   0]
       [ 18   0   0   0]
       [ 11   0 388   0]
       [ 26   0   0   0]]

[35]: from sklearn.metrics import classification_report

      print(classification_report(y_test, y_pred_gini))

                    precision    recall  f1-score   support

               acc       0.56      0.56      0.56       127
              good       0.00      0.00      0.00        18
             unacc       0.87      0.97      0.92       399
             vgood       0.00      0.00      0.00        26

          accuracy                           0.81       570
         macro avg       0.36      0.38      0.37       570
      weighted avg       0.74      0.81      0.77       570
```

Github :- https://github.com/Pranav-Divekar/Machine-learning-

Conclusion:

Decision Trees are effective for classification and regression but can overfit with deep trees. Pruning and hyperparameter tuning help improve performance. They provide a strong foundation for machine learning and are widely used in real-world applications.