**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

Program: B.Tech\MBA.Tech Computer Sem VI

**Course: Machine Learning**

Experiment No.09

PART A

**A.1 Aim: To implement Decision Tree for classification task**

**Task:**

1. **Upload car evaluation dataset into the data-frame**
2. **Apply Data exploration steps.**
3. **Apply missing value treatment if required.**
4. **Split data set into train test at a ratio of 80- 20**
5. **Apply ordinal encoder on columns like [‘buying’,’maint’,’doors’,’persons’,’lug\_boot’,’safety’]**
6. **Apply Decision Tree classifier with criterion as ‘entropy’ and maximum depth of the tree as 3**
7. **Predict class for test data set find accuracy of the model for test data set**
8. **Predict class for train data set and find accuracy of the model for the train data set**
9. **Compare accuracy of train and test data set and comment on overfitting or under fitting of the model**
10. **Plot decision tree using matplotlib**
11. **Identify the number of pure partitions (leaf nodes) created. Comment on the entropy of pure partitions.**
12. **Plot decision tree using graphviz library. Which attribute is selected as root node?**
13. **Plot the confusion matrix for test data set**
14. **Plot the classification report for the test data set. Comment your observation for classification report**

**A.2 Prerequisite:**

Python Programming, Numpy

**Theory:**

A decision tree classifier is a popular machine learning algorithm used for classification tasks. It operates by recursively partitioning the feature space into smaller regions and assigning a class label to each region. At each step, the decision tree algorithm selects the feature that best splits the data into homogeneous subsets regarding the target variable.

some common types of decision tree classifiers:

1. Binary Decision Trees: These are decision trees where each internal node has exactly two children. The splitting decision at each node involves dividing the data into two subsets based on a threshold value for a specific feature.

2. Multiway Decision Trees: In contrast to binary decision trees, multiway decision trees allow nodes to have more than two children. This enables more complex decision-making processes and can potentially result in more accurate models, especially for datasets with many classes or complex relationships.

3. Classification Trees: Classification trees are used when the target variable is categorical or qualitative. The tree is built to predict the class label of new instances based on the features provided.

4. Regression Trees: Regression trees are used when the target variable is continuous or quantitative. Instead of predicting a class label, these trees predict a numerical value for the target variable.

5. CART (Classification and Regression Trees): CART is a versatile algorithm that can be used for both classification and regression tasks. It constructs binary trees by recursively partitioning the data into subsets based on the feature that maximizes the information gain or minimizes the impurity at each step.

6. ID3 (Iterative Dichotomiser 3): ID3 is an algorithm specifically designed for building decision trees for classification tasks. It uses entropy and information gain measures to select the best splitting feature at each node.

7. C4.5: C4.5 is an extension of ID3 that addresses some of its limitations, such as handling continuous-valued attributes and missing data. It also introduces pruning techniques to avoid overfitting.

8. Random Forest: Random Forest is an ensemble learning technique that uses multiple decision trees to improve predictive accuracy and reduce overfitting. It constructs a forest of decision trees by training each tree on a random subset of the training data and aggregating their predictions.

These are some of the common types of decision tree classifiers. The choice of which type to use depends on the nature of the data and the specific requirements of the classification task.

**ID3 Decision Tree Classifier:**

ID3 (Iterative Dichotomiser 3) is a classic algorithm for constructing decision trees, primarily used for classification tasks. It was developed by Ross Quinlan in the 1980s and remains one of the fundamental algorithms in machine learning.

1. Entropy and Information Gain:

- ID3 uses the concept of entropy to measure the impurity or randomness in a dataset. Entropy is a measure of uncertainty in a random variable.

- For a given dataset, the entropy (H) is calculated using the formula:



Where S is the dataset, c is the number of classes, and pi is the proportion of instances in class *i* in the dataset.

- Information Gain is a measure of the effectiveness of a particular attribute in classifying the data. It quantifies the reduction in entropy achieved after splitting the dataset based on a specific attribute.

- Information Gain (IG) for a given attribute A is calculated as:



where S is the original dataset, A is the attribute being considered, *Values(A)* are the possible values of attribute A, Sv is the subset of S for which attribute A has value v, and |S| denotes the number of instances in S.

2. Building the Tree:

- ID3 follows a recursive approach to build the decision tree. At each step, it selects the attribute that maximizes the Information Gain.

- It starts with the root node and selects the attribute that provides the highest Information Gain to split the dataset.

- The dataset is then partitioned into subsets based on the values of the selected attribute.

- This process is recursively applied to each subset, building the tree from top to bottom until one of the termination conditions is met.

3. Termination Conditions:

- ID3 stops splitting the dataset and creates leaf nodes under the following conditions:

- All instances in the subset belong to the same class, making it a pure node.

- There are no remaining attributes to split on (i.e., the dataset cannot be further divided).

- A pre-defined stopping criterion, such as a maximum tree depth or minimum number of instances per leaf, is met.

4. Handling Categorical Attributes:

- ID3 is specifically designed for categorical attributes. It can handle both binary and multi-class classification problems.

- For attributes with multiple categorical values, ID3 considers each value separately when calculating Information Gain.

5. Pruning:

- ID3 does not perform pruning, which can lead to overfitting. It tends to create overly complex trees, especially when dealing with noisy or high-dimensional data.

ID3 is a simple yet powerful algorithm for building decision trees. However, it has some limitations, such as its inability to handle continuous attributes directly and its tendency to overfit noisy data. Variants like C4.5 and C5.0 address some of these limitations and extend the capabilities of the original ID3 algorithm.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical.)***

|  |  |
| --- | --- |
| Roll No. N052 | Name: Pratyush Kumar |
| Class : MBA Tech CE (div. D) | Batch : B2 |
| Date of Experiment: 09-03-2024 | Date of Submission: 10-03-2024 |
| Grade : |  |

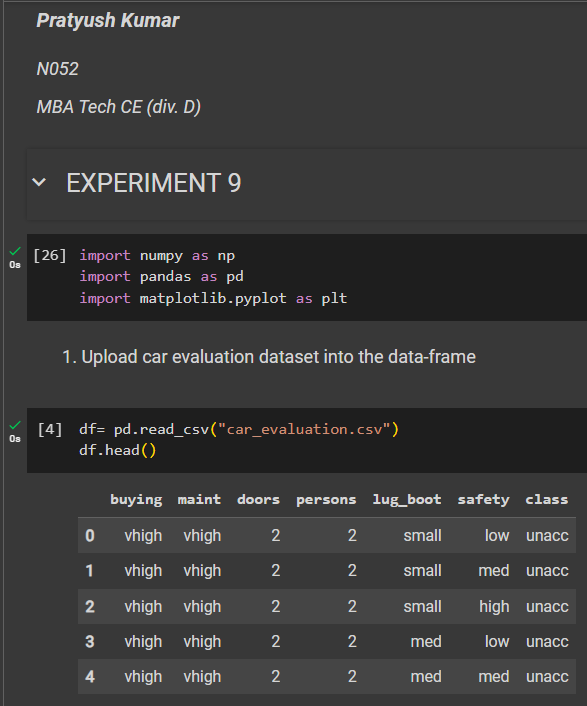
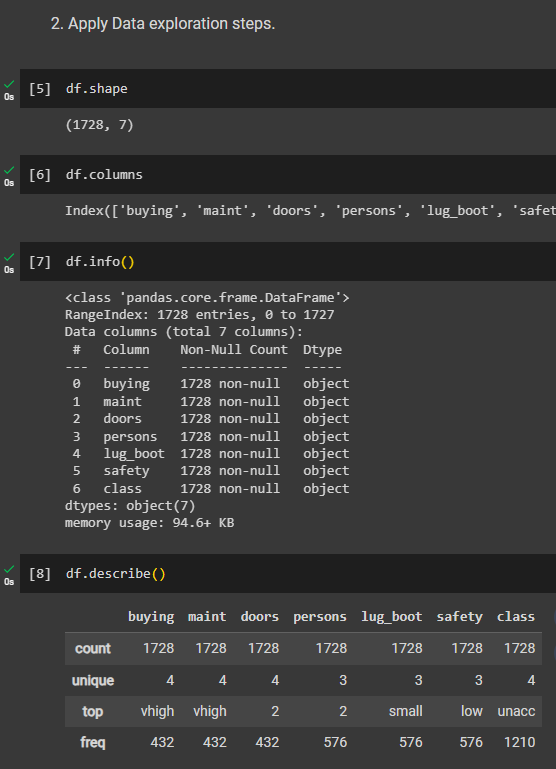
**B.1 Task 1**

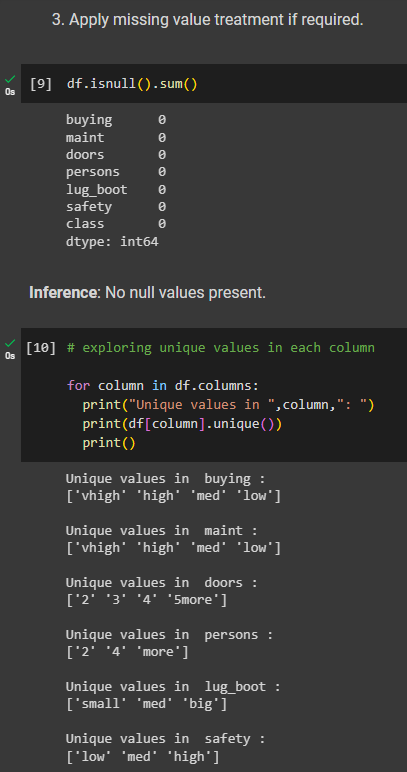
**Colab link:** https://colab.research.google.com/drive/1oB4Q8m4-0ZzkoAF1sjFrDME0sU59QCRg?usp=sharing

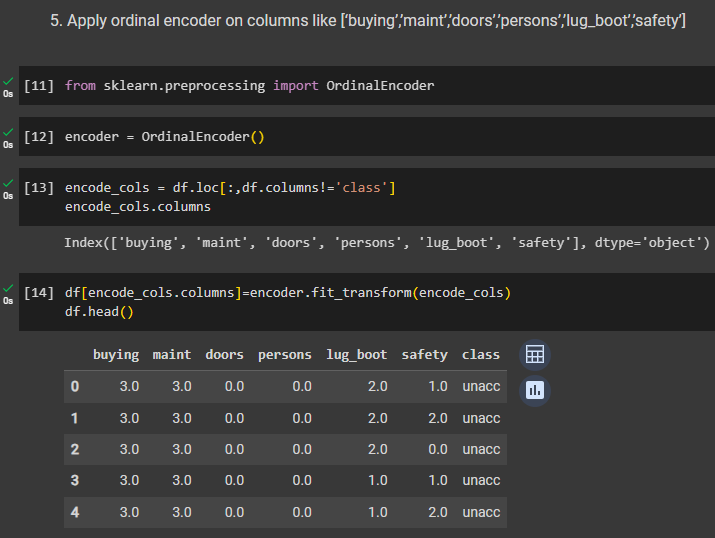
* **Source Code**

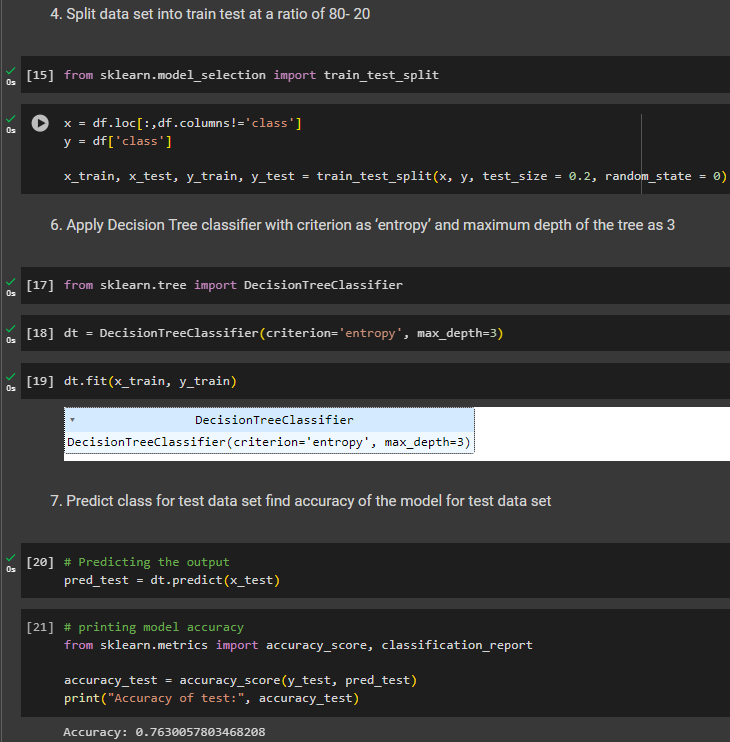
*"""  
 \* This file contains code snippets to implement decision tree classifier on car\_evaluation dataset  
 \* ML-E9-Task1  
 \*  
 \* Original file is located at: https://colab.research.google.com/drive/1oB4Q8m4-0ZzkoAF1sjFrDME0sU59QCRg  
 \* @author Pratyush Kumar (github.com/pratyushgta)  
"""*"""  
## EXPERIMENT 9  
"""  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
"""1. Upload car evaluation dataset into the data-frame"""  
  
df= pd.read\_csv("car\_evaluation.csv")  
df.head()  
  
"""2. Apply Data exploration steps."""  
  
df.shape  
  
df.columns  
  
df.info()  
  
df.describe()  
  
"""3. Apply missing value treatment if required."""  
  
df.isnull().sum()  
  
"""\*\*Inference\*\*: No null values present."""  
  
# exploring unique values in each column  
for column in df.columns:  
 print("Unique values in ",column,": ")  
 print(df[column].unique())  
 print()  
  
"""5. Apply ordinal encoder on columns like [‘buying’,’maint’,’doors’,’persons’,’lug\_boot’,’safety’]"""  
  
from sklearn.preprocessing import OrdinalEncoder  
  
encoder = OrdinalEncoder()  
  
encode\_cols = df.loc[:,df.columns!='class']  
encode\_cols.columns  
  
df[encode\_cols.columns]=encoder.fit\_transform(encode\_cols)  
df.head()  
  
"""4. Split data set into train test at a ratio of 80- 20"""  
  
from sklearn.model\_selection import train\_test\_split  
  
x = df.loc[:,df.columns!='class']  
y = df['class']  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 0)  
  
"""6. Apply Decision Tree classifier with criterion as ‘entropy’ and maximum depth of the tree as 3"""  
  
from sklearn.tree import DecisionTreeClassifier  
  
dt = DecisionTreeClassifier(criterion='entropy', max\_depth=3)  
  
dt.fit(x\_train, y\_train)  
  
"""7. Predict class for test data set find accuracy of the model for test data set"""  
  
# Predicting the output  
pred\_test = dt.predict(x\_test)  
  
# printing model accuracy  
from sklearn.metrics import accuracy\_score, classification\_report  
  
accuracy\_test = accuracy\_score(y\_test, pred\_test)  
print("Accuracy of test:", accuracy\_test)  
  
"""8. Predict class for train data set and find accuracy of the model for the train data set"""  
  
# Predicting the output  
pred\_train = dt.predict(x\_train)  
  
# printing model accuracy  
accuracy\_train = accuracy\_score(y\_train, pred\_train)  
print("Accuracy of train:", accuracy\_train)  
  
"""9. Compare accuracy of train and test data set and comment on overfitting or under fitting of the model"""  
# Comparing the accuracy of train and test data sets  
if accuracy\_train > accuracy\_test:  
 print("The model is overfitting")  
elif accuracy\_train < accuracy\_test:  
 print("The model is underfitting")  
else:  
 print("Neither overfitting nor underfitting")  
  
"""10. Plot decision tree using matplotlib"""  
  
from sklearn import tree  
  
plt.figure(figsize=(25, 10))  
tree.plot\_tree(dt, feature\_names=x\_train.columns, class\_names=df['class'].unique(), fontsize=14, filled=True)  
plt.show()  
  
"""11. Identify the number of pure partitions (leaf nodes) created. Comment on the entropy of pure partitions."""  
  
# get number of leaf nodes  
num\_leaf\_nodes = dt.get\_n\_leaves()  
print("Number of pure partitions (leaf nodes):", num\_leaf\_nodes)  
  
"""\*\*Comment on entropy:\*\* The entropy of a pure partition is 0, because there is only one class present in the partition.  
  
12. Plot decision tree using graphviz library. Which attribute is selected as root node?  
"""  
  
import graphviz  
  
dt\_data = tree.export\_graphviz(dt, out\_file=None, feature\_names=x\_train.columns, class\_names=df['class'].unique(), filled=True, rounded=True, special\_characters=True)  
graphviz.Source(dt\_data)  
  
"""13. Plot the confusion matrix for test data set"""  
  
from sklearn.metrics import accuracy\_score, confusion\_matrix, ConfusionMatrixDisplay, f1\_score  
  
# Computing confusion matrix  
#labels = [0,1]  
cm = confusion\_matrix(y\_test, pred\_test)  
#print(cm)  
  
# Plotting confusion matrix  
display\_cm = ConfusionMatrixDisplay(confusion\_matrix=cm)#display\_labels=labels)  
display\_cm.plot();  
  
"""14. Plot the classification report for the test data set. Comment your observation for classification report"""  
  
print(classification\_report(y\_test, pred\_test))  
  
"""\*\*Observations:\*\*  
1. The classification report shows that the model has a good performance for the test data set.  
2. Precision, recall and f1-score for each class are all above 0.7. This indicates that the model is able to correctly classify most of the instances in the test data set.  
"""

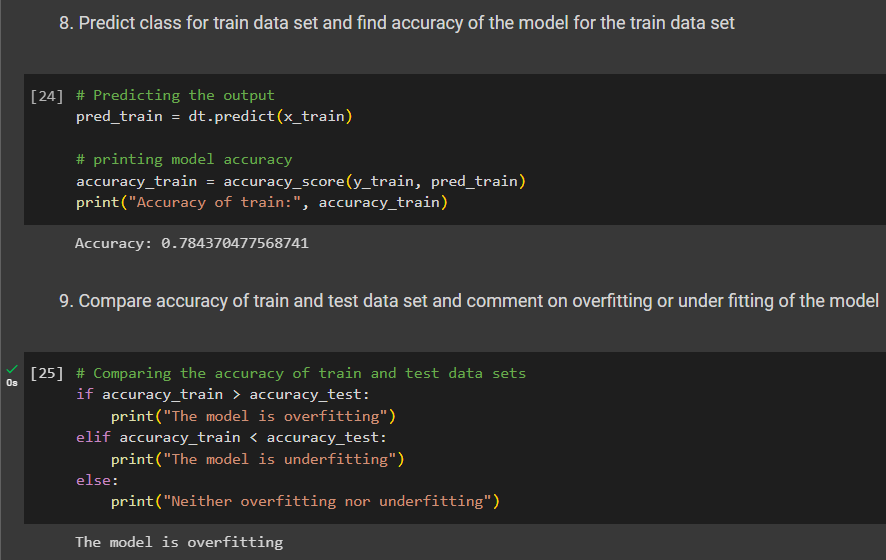
* **Input/ Output**

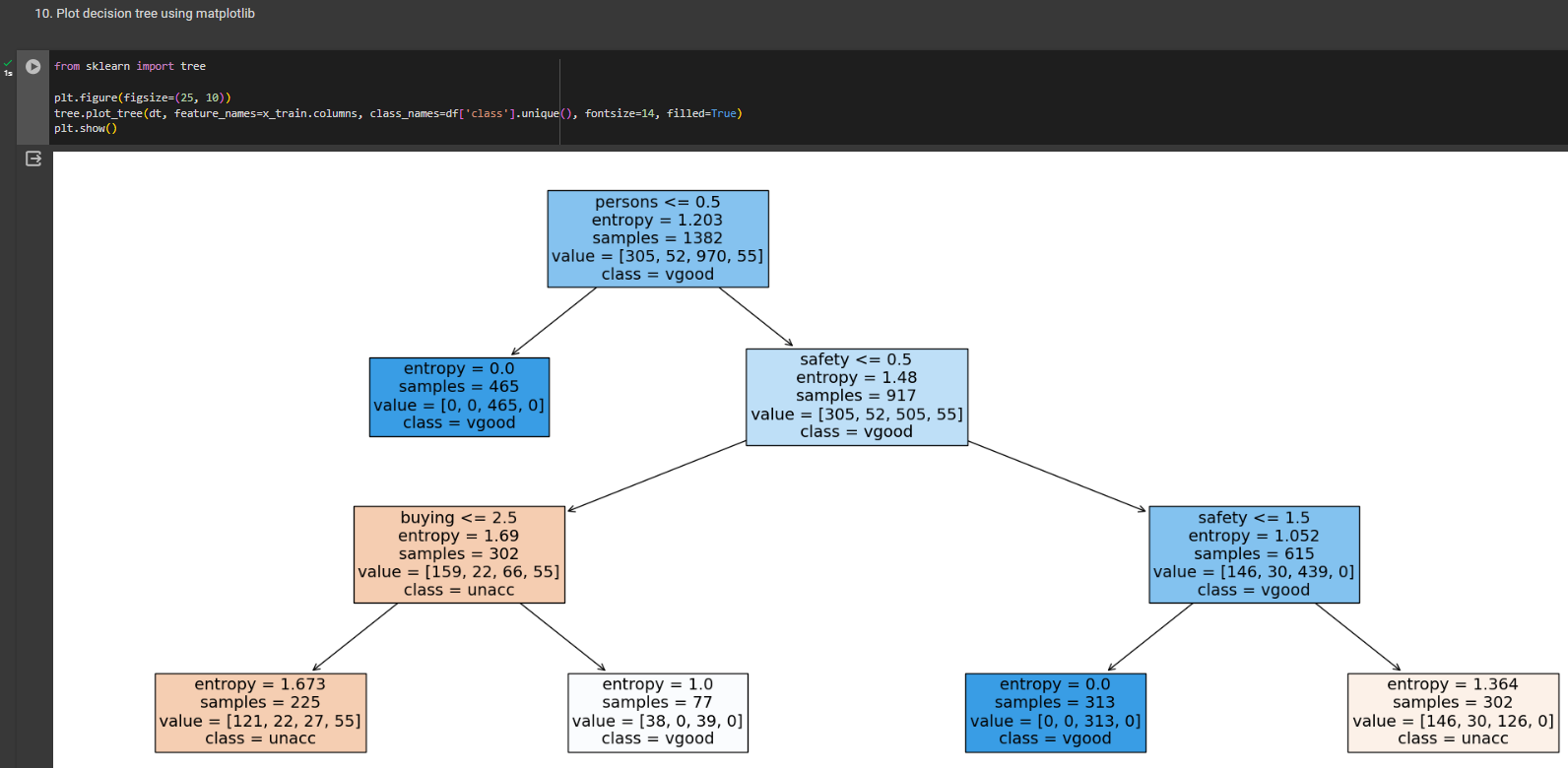
** **

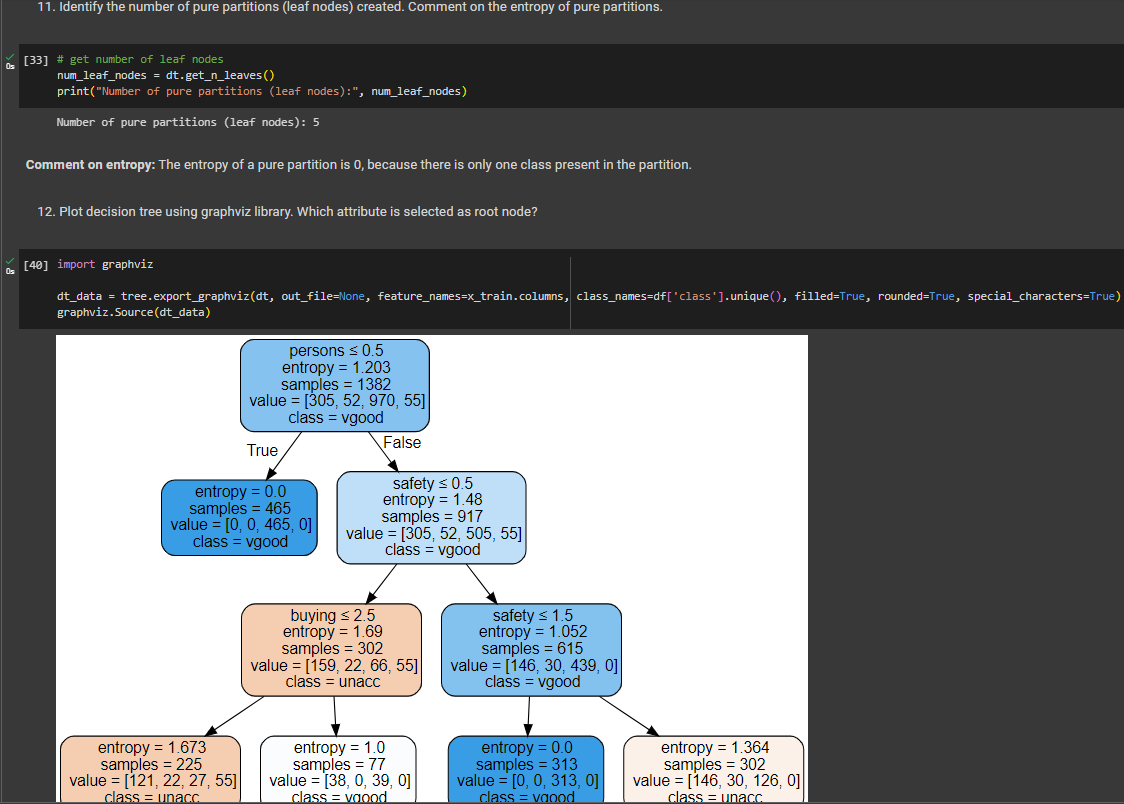
****

****

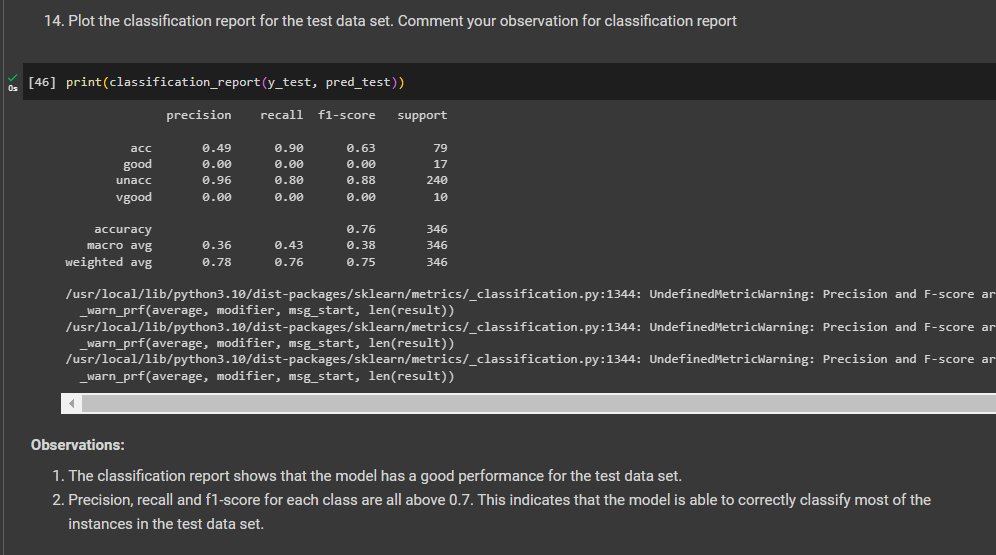
****

****

****

****

****

****

**B.2 Conclusion:**

*(Students must write the conclusion in their own words.)*

Performed Decision Tree classification on car\_evaluation dataset. It was successfully able to classify the data with a high degree of accuracy, enabling it to efficiently handle both categorical and numerical data. However, it was observed that the model could be prone to overfitting if not properly tuned.