Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Ayush Jain SAP ID: 60004200182

Course: Machine Learning Laboratory Course Code: **DJ19CEEL6021**

Year: **T.Y. B.Tech.** Sem: **VII** Batch: B 3

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	2, 4	2, 4	2, 4	2, 4	2, 4	3	2, 4	2, 4	5								
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	4	5	5	5													
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	4	4	4	5													
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5													
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	4	4	4	4													
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–													
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	4	4	4	4													
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–													
Total	21	22	22	23													
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member:

Signature of Head of the Department
Date:

PLOT NO. U-15, JVPD SCHEME, BHAKTIVEDANTA SWAMI MARG, VILE PARLE (WEST), MUMBAI - 400056
Tel.: 42335000/42335001 Email : info@djsce.ac.in / admin@djsce.ac.in Website : www.djsce.ac.in

Machine Learning

Experiment 3

Ayush Jain 60004200132 B3

Aim : To implement CART in python without using any inbuilt function and with inbuilt function.

Theory:

Introduction:
CART (Classification and Regression Tree) is a decision tree algorithm that can be used for both classification and regression tasks. In regression problems, CART creates a binary tree that recursively splits the data into subsets based on the value of a selected feature until a stopping criterion is met. The algorithm works by evaluating the impurity of each split and selecting the feature that produces the purest subsets possible.

Gini Index:
The Gini index is a metric used to evaluate the quality of a split in the CART algorithm. It measures the probability of incorrectly classifying a randomly chosen element from the set. In the CART algorithm, the Gini index is used to evaluate the impurity of a node. A split is made on the feature that produces the lowest Gini index, resulting in the purest subsets possible.
The formula for Gini index is as follows:

Gini Index = 1 - ∑(pi)^2

Where pi is the probability of an element being classified to a particular class. The Gini index ranges from 0 to 1, with a value of 0 indicating that all elements in the node belong to the same class, and a value of 1 indicating that the elements are uniformly distributed across all classes.

CART Algorithm for Classification
Here is the approach for most decision tree algorithms at their most simplest. The tree will be constructed in a top-down approach as follows:
Step 1: Start at the root node with all training instances
Step 2: Select an attribute on the basis of splitting criteria (Gain Ratio or other impurity metrics, discussed below)
Step 3: Partition instances according to selected attribute recursively

Partitioning stops when:

- There are no examples left
- All examples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority class is the leaf

Example:

Let's start with a simple example. Assume you have a bunch of oranges and mandrins with labels on them, and you want to identify a set of simple rules that you can use in the future to distinguish between these two types of fruit.

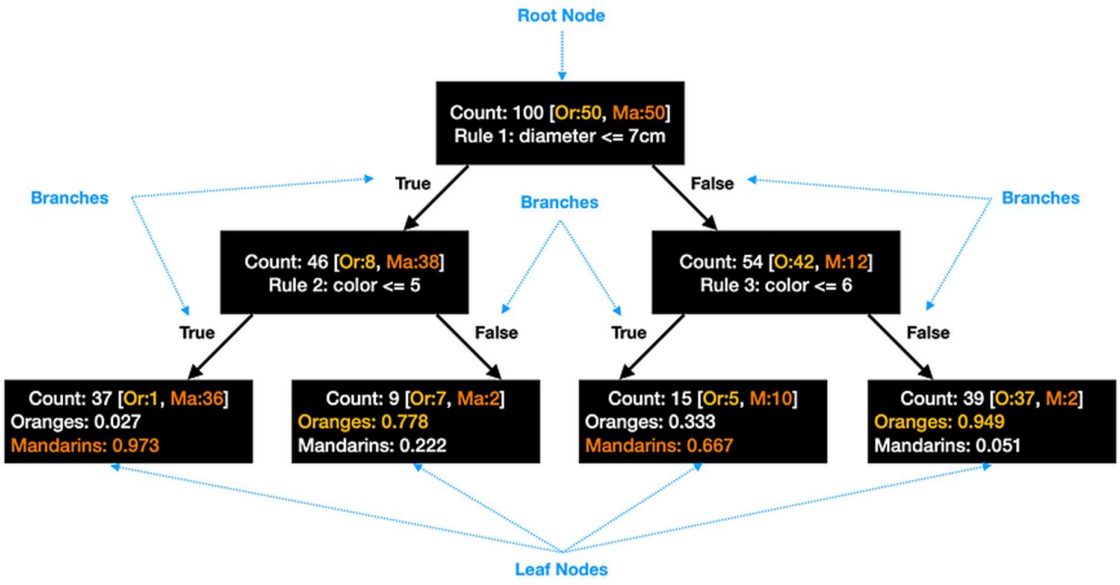
Typically, oranges (diameter 6–10cm) are bigger than mandarins (diameter 4–8cm), so the first rule found by your algorithm might be based on size:

- Diameter ≤ 7cm.

Next, you may notice that mandarins tend to be slightly darker in color than oranges. So, you use a color scale (1=dark to 10=light) to split your tree further:

- Color ≤5 for the left side of the sub-tree
- Color ≤6 for the right side of the sub-tree

Your final result is a tree that consists of 3 simple rules that help you to correctly distinguish between oranges and mandarins in the majority of the cases:



Code without inbuilt function:

```
import pandas as pd
import numpy as np
```

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

# Define a function to calculate the Gini index for a given attribute value
def calc_gini_for_attribute(class_name, col, df, target_col='Buys_computer'):
    total_count = len(df[df[col].isin([class_name])])
    count_of_1 = len(df[(df[col].isin([class_name])) & (df[target_col] == 1)])
    count_of_0 = len(df[(df[col].isin([class_name])) & (df[target_col] == 0)])
    prob_of_1 = count_of_1 / total_count
    prob_of_0 = count_of_0 / total_count
    gini = 1 - (prob_of_1 **2) - (prob_of_0**2)
    return gini, total_count

# Define a function to calculate the Gini index for all attributes
def calc_gini(cols: list, data, target_col='Buys_computer'):
    gini_dict = {}
    for col in cols:
        gini_for_attr = 0
        for value in list(data[col].unique()):
            gini_val, var_count = calc_gini_for_attribute(value, col, data)
            gini_for_attr += var_count/len(data) * gini_val
        gini_dict[col] = round(gini_for_attr, 3)
    return gini_dict

# Convert the "Buys_computer" feature to a categorical variable
df["Buys_computer"] = pd.Categorical(df["Buys_computer"])

# Get the unique classes of the "Buys_computer" feature
classes = df["Buys_computer"].unique()

# Loop through each attribute in the dataset
for attribute in df.columns[:-1]:
    # Convert the attribute to a categorical variable
    df[attribute] = pd.Categorical(df[attribute])
    # Calculate the Gini index for the attribute
    gini_dict = calc_gini([attribute], df, target_col='Buys_computer')
    # Print the Gini index for the attribute
    print("Gini Index of", attribute + ":", gini_dict[attribute])
```

Output:

```
Gini Index of Age: 1.0
Gini Index of Income: 1.0
Gini Index of Student: 1.0
Gini Index of Credit_rating: 1.0
Gini Index of Buys_computer: 1.0
```

Code with inbuilt function:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn import tree
from sklearn import metrics

pd.options.display.max_columns=50
df=pd.read_csv('weatherAUS.csv', encoding='utf-8')
df=df[pd.isnull(df['RainTomorrow'])==False]
df=df.fillna(df.mean())
df['RainTodayFlag']=df['RainToday'].apply(lambda x: 1 if x=='Yes' else 0)
df['RainTomorrowFlag']=df['RainTomorrow'].apply(lambda x: 1 if x=='Yes'
else 0)

X=df[['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustS
peed',
    'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressur
e9am',
    'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday
Flag']]
y=df['RainTomorrowFlag'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
ate=0)

model = tree.DecisionTreeClassifier(criterion="gini",
                                    splitter="best")
clf = model.fit(X_train, y_train)


train = model.predict(X_train)
print(train)
test = model.predict(X_test)
```

```
print(test)
print("Training accuracy : ",metrics.accuracy_score(y_train , train))
print("Testing accuracy  : ",metrics.accuracy_score(y_test , test))
```

Output:

```
[0 0 0 ... 0 0 1]
[0 0 0 ... 0 0 1]
Training accuracy :  0.9999648364013574
Testing accuracy  :  0.7862090790815429
```

Conclusion: We implemented CART in python without using any inbuilt function and with inbuilt function.