**Decision Tree**

**CSE 303: Machine Learning**

Submitted by

Name: Roda Chinthapalli

Roll No: AP22110011496

Section :M

Lab Bate: 10/09/24

End Date: 16/09/24

**A picture containing text

Description automatically generated**

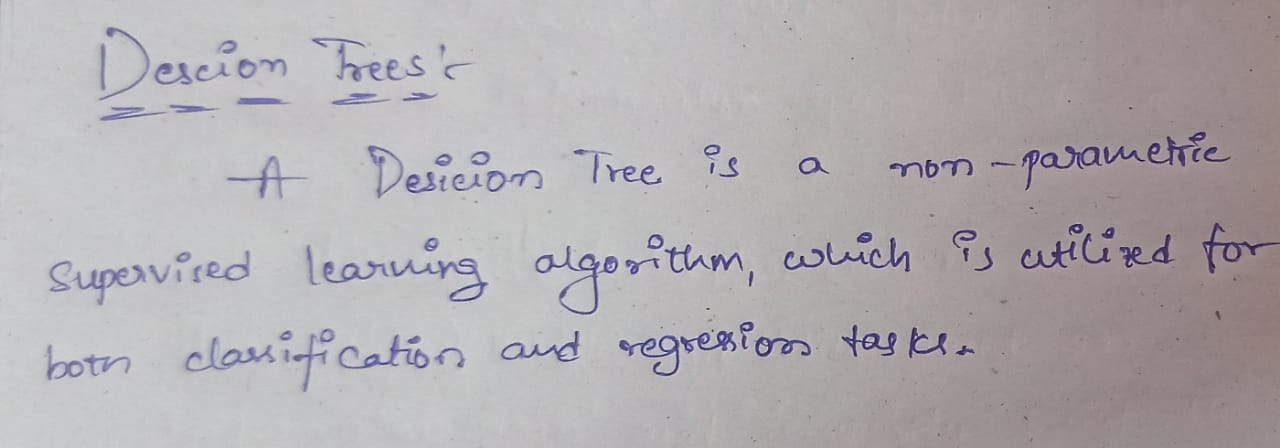
**Department Computer Science and Engineering**

**School of Engineering and Sciences**

**SRM University–AP**

**Amaravati, Andhra Pradesh – 522 240, India**

**DECISION TREE**

****

**1.Question:**

Implement Decision Tree Classifier for classification of EnjoySport dataset

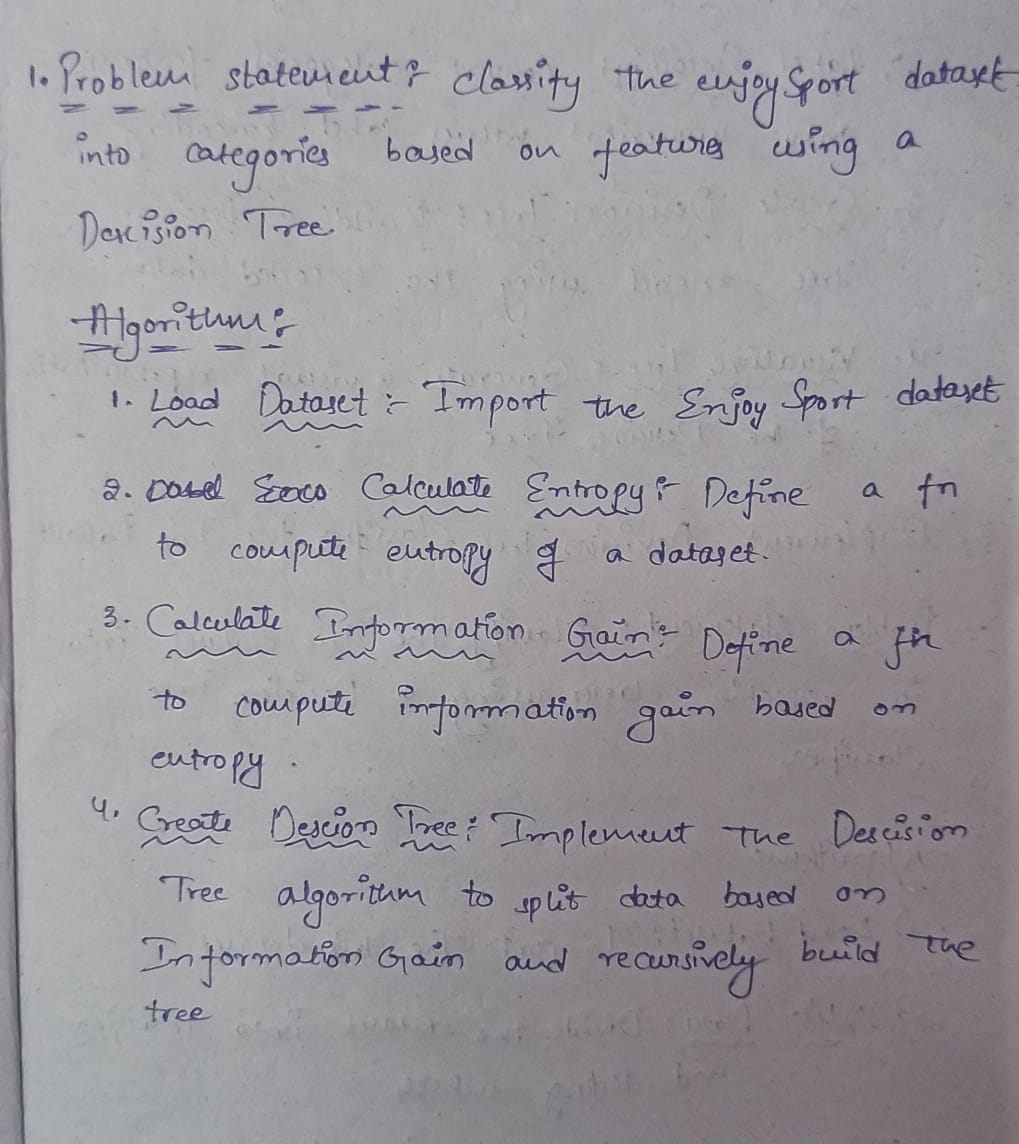
a. Load the data set

b. Create a function for calculating entropy

c. Create a function for calculating InformationGain

d. write code for creating Discission Tree

**Problem Statement and Algorithm:**

****

**Solution:**

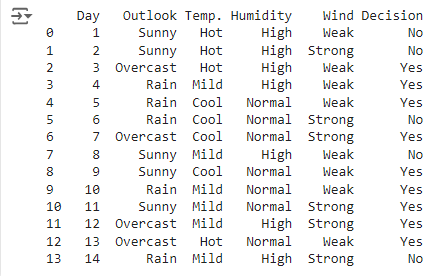
import pandas as pd

# Load the dataset

df = pd.read\_csv('/content/Enjoy sports.csv')

# Display the entire dataset

print(df)

****

**a.Create a function for calculating entropy**

import numpy as np

import pandas as pd

def calculate\_entropy(column):

    # Calculate frequency of each unique value in the column

    values, counts = np.unique(column, return\_counts=True)

    probabilities = counts / len(column)

    # Calculate the entropy

    entropy = -np.sum(probabilities \* np.log2(probabilities))

    return entropy

# Load the dataset

df = pd.read\_csv('/content/Enjoy sports.csv')

# List of columns to compute entropy for

columns\_to\_check = ['Outlook', 'Temp.', 'Humidity', 'Wind', 'Decision']

# Calculate and display entropy for each column

for column in columns\_to\_check:

    if column in df.columns:

        entropy\_value = calculate\_entropy(df[column])

        print(f"Entropy of '{column}': {entropy\_value}")

    else:

        print(f"Column '{column}' does not exist in the DataFrame.")

# Specific analysis for categorical columns

categorical\_columns = ['Outlook', 'Temp.', 'Humidity', 'Wind']

for column in categorical\_columns:

    if column in df.columns:

        unique\_values = df[column].unique()

        for value in unique\_values:

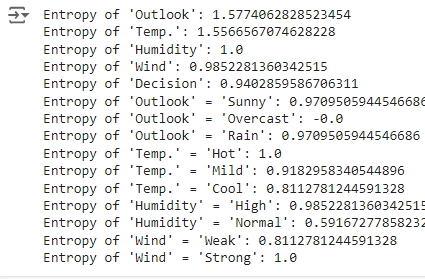
            filtered\_df = df[df[column] == value]

            entropy\_value = calculate\_entropy(filtered\_df['Decision'])

            print(f"Entropy of '{column}' = '{value}': {entropy\_value}")

    else:

        print(f"Column '{column}' does not exist in the DataFrame.")

****

**C.Create a function for calculating Information Gain**

import numpy as np

import pandas as pd

def calculate\_entropy(column):

    # Calculate frequency of each unique value in the column

    values, counts = np.unique(column, return\_counts=True)

    probabilities = counts / len(column)

    # Calculate the entropy

    entropy = -np.sum(probabilities \* np.log2(probabilities))

    return entropy

def calculate\_information\_gain(df, attribute, target):

    # Calculate the entropy of the original dataset

    original\_entropy = calculate\_entropy(df[target])

    # Calculate the weighted entropy after the split

    values, counts = np.unique(df[attribute], return\_counts=True)

    weighted\_entropy = 0

    total\_count = len(df)

    for value, count in zip(values, counts):

        subset = df[df[attribute] == value]

        subset\_entropy = calculate\_entropy(subset[target])

        weighted\_entropy += (count / total\_count) \* subset\_entropy

    # Calculate information gain

    information\_gain = original\_entropy - weighted\_entropy

    return information\_gain

# Load the dataset

df = pd.read\_csv('/content/Enjoy sports.csv')

# Calculate Information Gain for a specific attribute

attributes = ['Outlook', 'Temp.', 'Humidity', 'Wind']

for attribute in attributes:

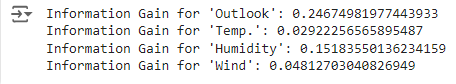
    if attribute in df.columns:

        info\_gain = calculate\_information\_gain(df, attribute, 'Decision')

        print(f"Information Gain for '{attribute}': {info\_gain}")

    else:

        print(f"Column '{attribute}' does not exist in the DataFrame.")

****

**d. Write a code for creation Decision Tree**

import pandas as pd

import numpy as np

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('/content/Enjoy sports.csv')

# Set pandas display options to show all columns and rows

pd.set\_option('display.max\_columns', None)  # Show all columns

pd.set\_option('display.max\_rows', None)     # Show all rows

pd.set\_option('display.max\_colwidth', None) # Show full column width

pd.set\_option('display.expand\_frame\_repr', False) # Prevent truncation in wide DataFrames

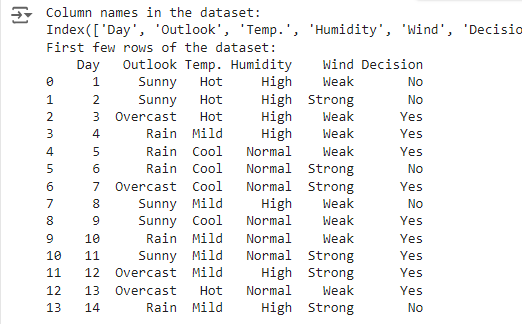
# Print column names and first few rows to check for issues

print("Column names in the dataset:")

print(df.columns)

print("First few rows of the dataset:")

print(df)

****

# Define the target column based on your dataset

target\_column = 'Decision'

# Ensure the target column is present

if target\_column not in df.columns:

    raise KeyError(f"The column '{target\_column}' is not found in the dataset")

# Prepare the data

X = df.drop(columns=[target\_column])

y = df[target\_column]

# Convert categorical features to numerical values

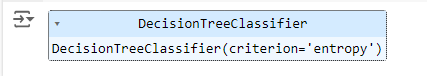
X\_encoded = pd.get\_dummies(X)

# Initialize the Decision Tree Classifier

clf = DecisionTreeClassifier(criterion='entropy')

# Fit the model

clf.fit(X\_encoded, y)

****

# Display the decision tree

def display\_tree(clf, feature\_names):

    plt.figure(figsize=(20,10))

    plot\_tree(clf,

              feature\_names=feature\_names,

              class\_names=clf.classes\_,

              filled=True,

              rounded=True,

              fontsize=12)

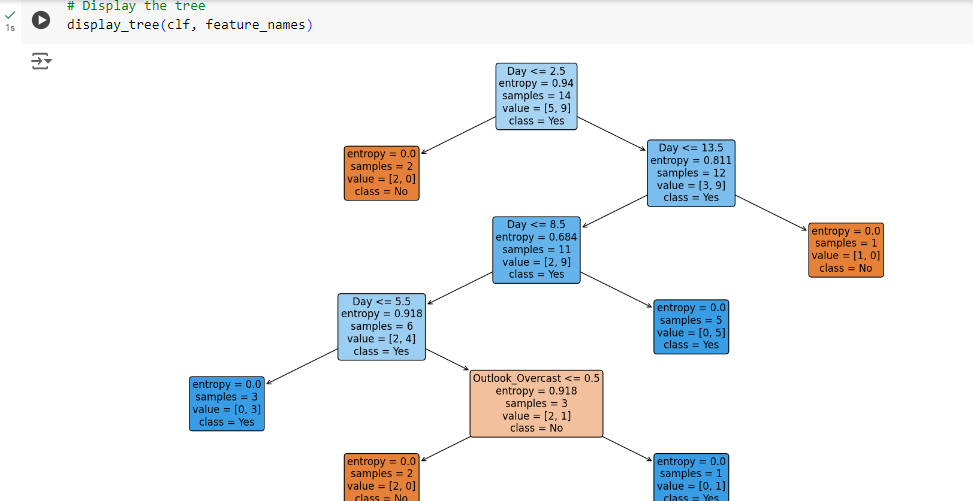
    plt.show()

# Get feature names after encoding

feature\_names = X\_encoded.columns

# Display the tree

display\_tree(clf, feature\_names)



**2.Question**

Implement Decision Tree Classifier for classification of EnjoySport dataset

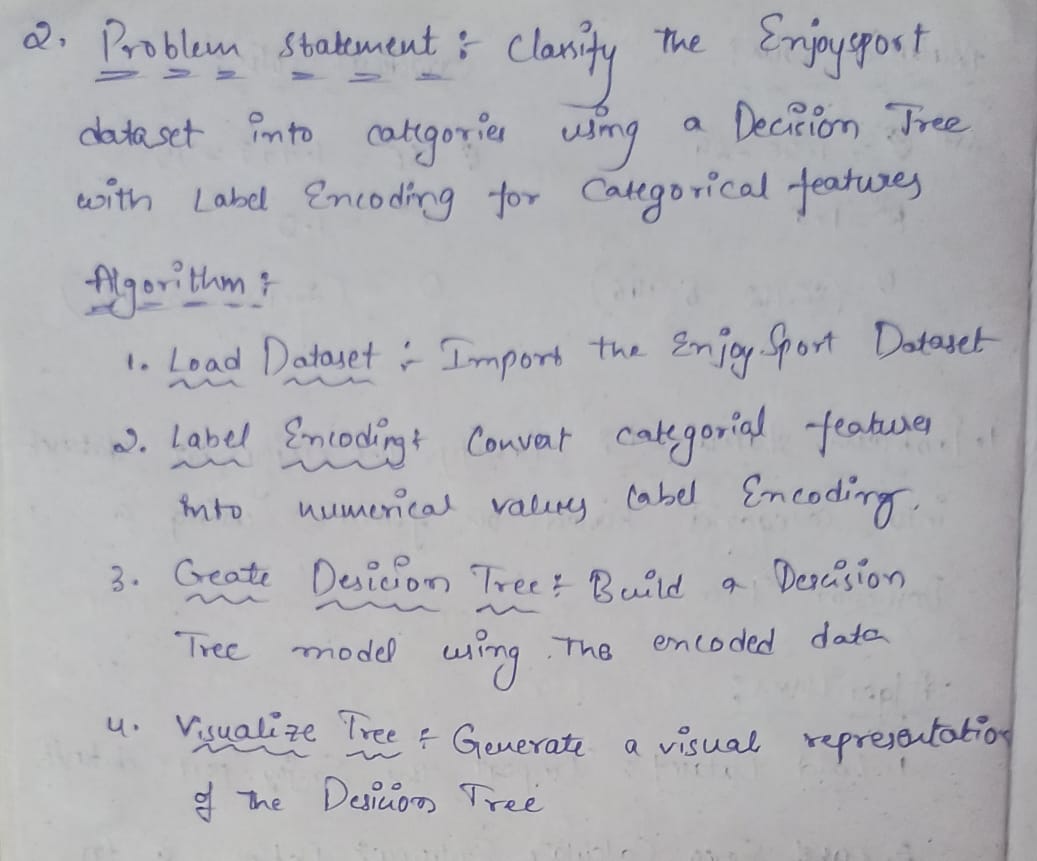
a. Load the data set

b. Use Label Encoding

c. create a Decision Tree

d. Visualize the tree

**Problem Statement and Algorithm:**

****

**Solution:**

**a.Load the Dataset**

import pandas as pd

# Load the dataset

file\_path = '/content/Enjoy sports.csv'

data = pd.read\_csv(file\_path)

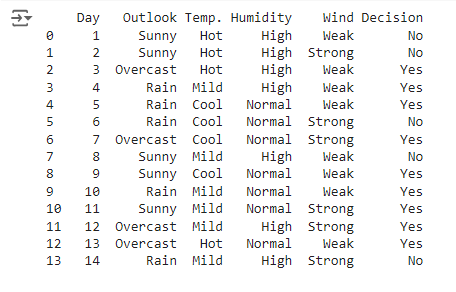
# show all rows and columns

pd.set\_option('display.max\_rows', None)

pd.set\_option('display.max\_columns', None)

# Display the entire DataFrame

print(data)

****

**b.Use Label Encoding**

from sklearn.preprocessing import LabelEncoder

import pandas as pd

# Initialize label encoders for each column

label\_encoders = {}

for column in data.columns:

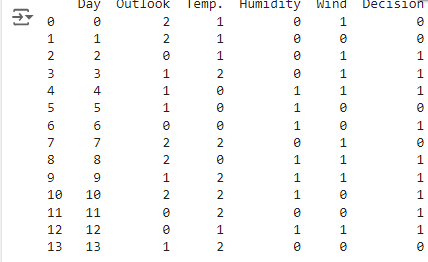
    le = LabelEncoder()

    data[column] = le.fit\_transform(data[column])

    label\_encoders[column] = le

# Display the entire transformed DataFrame

print(data)



**c.Create a Decision Tree:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Assuming `data` is your DataFrame and it has been properly loaded

# Separate features and target variable

X = data.drop('Decision', axis=1)  # Features

y = data['Decision']  # Target variable

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Initialize the Decision Tree Classifier

clf = DecisionTreeClassifier()

# Train the classifier

clf.fit(X\_train, y\_train)

# Make predictions

y\_pred = clf.predict(X\_test)

# Print accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')



**d. Visualize the Tree:**

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

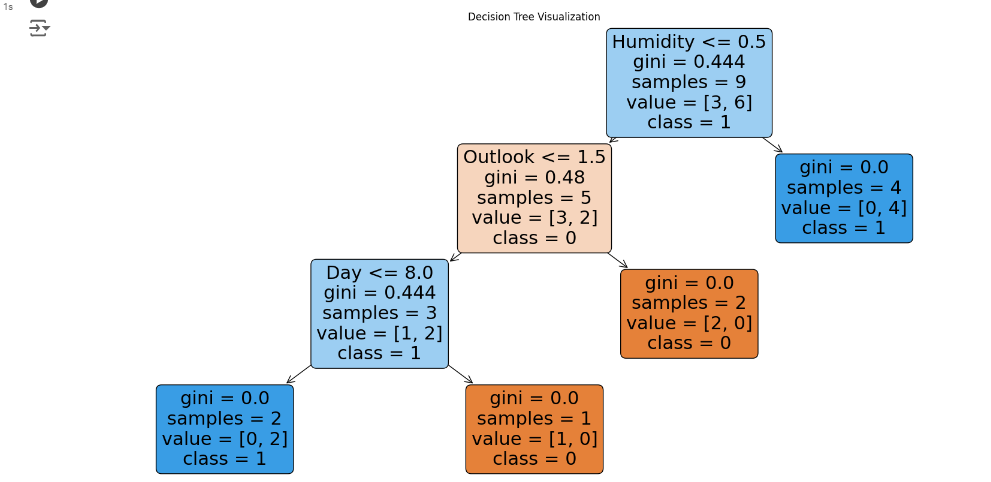
# Visualize the Decision Tree

plt.figure(figsize=(20,10))

plot\_tree(clf, filled=True, feature\_names=X.columns, class\_names=['0', '1'], rounded=True)

plt.title("Decision Tree Visualization")

plt.show()

****

**3.Question:**

Implement Decision Tree Classifier for classification of Iris dataset

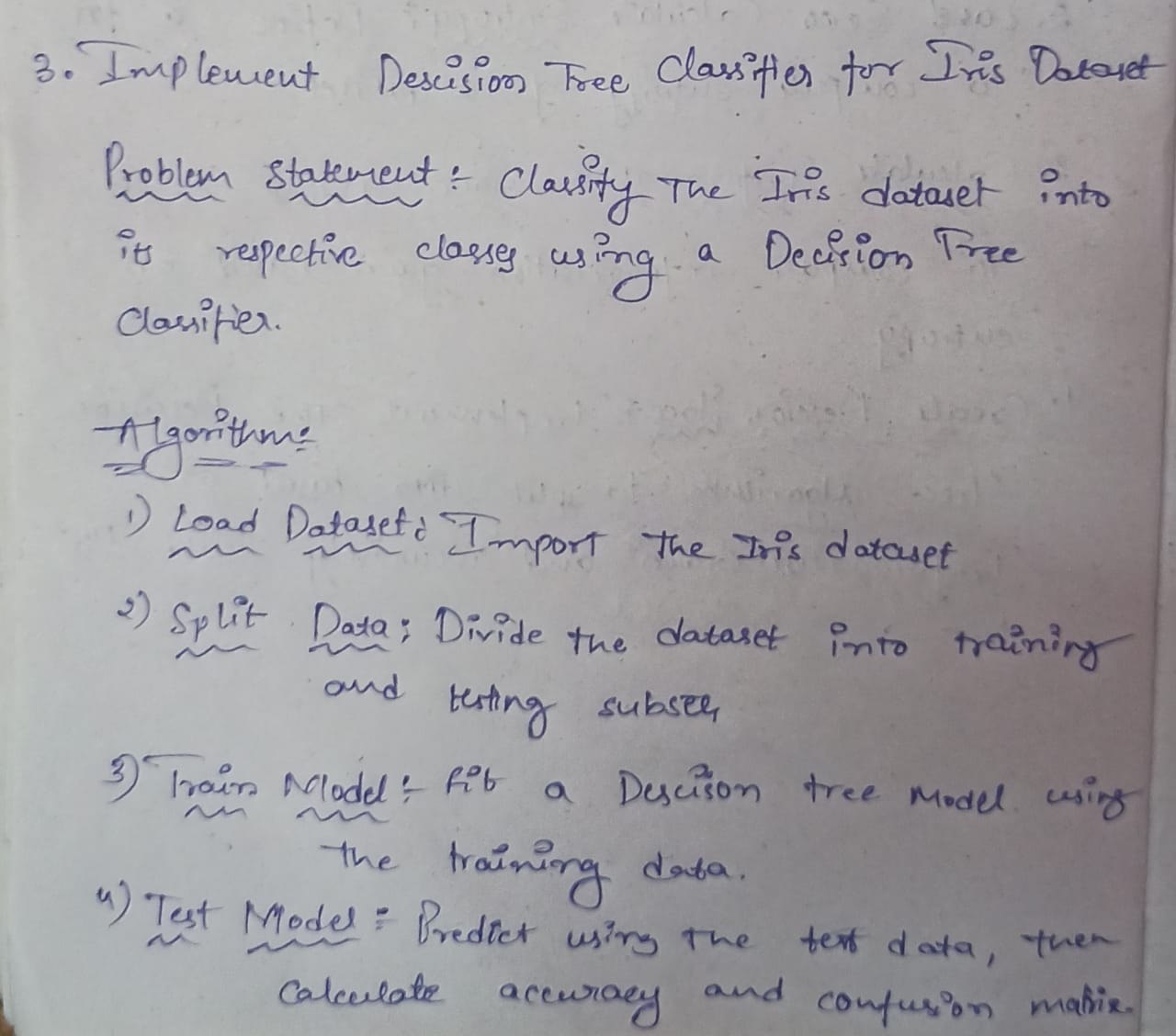
a. Load the data set

b. Split the data set to train and test sets

c. Train a Decision Tree using train set

d. Test the model using test set. Find accuracy and confusion Matrix.

**Problem Statement and Algorithm:**

****

**Solution:**

**a.Load the dataset:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

# Load the dataset

iris\_data = pd.read\_csv('/content/iris.csv')

# Define features and target

X = iris\_data.iloc[:, :-1]  # Features

y = iris\_data['Species']     # Target

**b.Split the data set to train and test sets**

# Split the data 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)

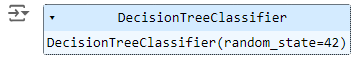
**c.Train a Decision Tree Calssifier:**

from sklearn.tree import DecisionTreeClassifier

# Create and train the Decision Tree classifier

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X\_train, y\_train)

****

**d.Test the model and find the accuracy and confusion matrix:**

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Make predictions

y\_pred\_dt = dt\_classifier.predict(X\_test)

# Calculate accuracy

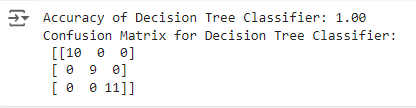
accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

print(f"Accuracy of Decision Tree Classifier: {accuracy\_dt:.2f}")

# Confusion matrix

conf\_matrix\_dt = confusion\_matrix(y\_test, y\_pred\_dt)

print("Confusion Matrix for Decision Tree Classifier:\n", conf\_matrix\_dt)

****

**4. Question :**

Implement Random Forest Classifier for classification of Iris dataset

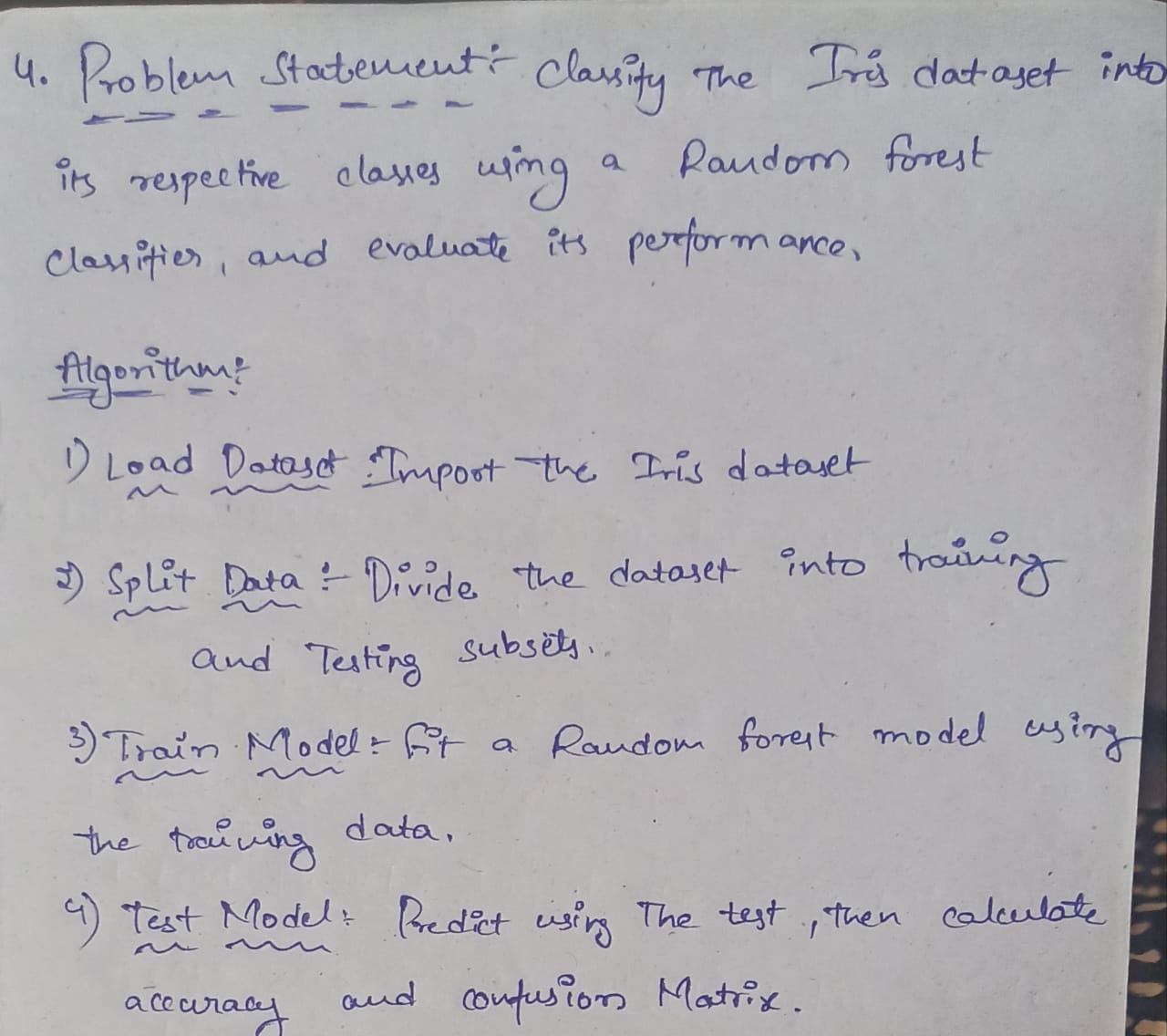
a. Load the data set

b. Split the data set to train and test sets

c. Train a Random Forest model using train set

d. Test the model using test set. Find accuracy and confusion Matrix.

**Problem Statement and Algorithm:**

****

**Solution:**

**a.Load the Dataset:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

# Load the dataset

iris\_data = pd.read\_csv('/content/iris.csv')

# Define features and target

X = iris\_data.iloc[:, :-1]  # Features

y = iris\_data['Species']     # Target

**b.Split the DAtaset into the Train and Test sets**

# Split the data 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)

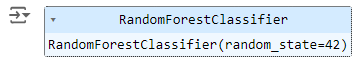
**c.Train a Random Forest Classifier**

from sklearn.ensemble import RandomForestClassifier

# Create and train the Random Forest classifier

rf\_classifier = RandomForestClassifier(random\_state=42)

rf\_classifier.fit(X\_train, y\_train)

****

**d.Test and model and Find accuracy and confusion matrix:**

# Make predictions

y\_pred\_rf = rf\_classifier.predict(X\_test)

# Calculate accuracy

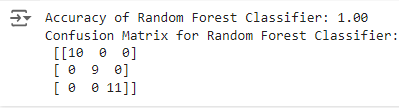
accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

print(f"Accuracy of Random Forest Classifier: {accuracy\_rf:.2f}")

# Confusion matrix

conf\_matrix\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

print("Confusion Matrix for Random Forest Classifier:\n", conf\_matrix\_rf)

****

**Code Repository:**

**https://github.com/Roda1458/Decision-Tree**