SET A 1. Write a Python program build Decision Tree Classifier using Scikit-learn package for diabetes data set (download database from <https://www.kaggle.com/uciml/pima-indiansdiabetes-database>)

ANS=

# Import required libraries

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

# Load the dataset

file\_path = "diabetes.csv" # Replace with your file path

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

# Display dataset structure

print("Dataset Info:")

print(data.info())

# Split the data into features (X) and target (y)

X = data.drop(columns=["Outcome"])

y = data["Outcome"]

# 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)

# Create and train the Decision Tree Classifier

clf = DecisionTreeClassifier(random\_state=42)

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

# Make predictions

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

# Evaluate the model

print("\nModel Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

# Visualize the Decision Tree

plt.figure(figsize=(15, 10))

plot\_tree(clf, feature\_names=X.columns, class\_names=["No Diabetes", "Diabetes"], filled=True)

plt.title("Decision Tree Visualization")

plt.show()

Dataset Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 19 entries, 0 to 18

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pregnancies 19 non-null int64

1 Glucose 19 non-null int64

2 BloodPressure 19 non-null int64

3 SkinThickness 19 non-null int64

4 Insulin 19 non-null int64

5 BMI 19 non-null float64

6 DiabetesPedigreeFunction 19 non-null float64

7 Age 19 non-null int64

8 Outcome 19 non-null int64

dtypes: float64(2), int64(7)

memory usage: 1.4 KB

None

Model Accuracy: 0.5

Classification Report:

precision recall f1-score support

0 0.50 0.50 0.50 2

1 0.50 0.50 0.50 2

accuracy 0.50 4

macro avg 0.50 0.50 0.50 4

weighted avg 0.50 0.50 0.50 4

2. Write a Python program build Decision Tree Classifier for shows.csv from pandas and predict class label for show starring a 40 years old American comedian, with 10 years of experience, and a comedy ranking of 7? Create a csv file as shown in [https://www.w3schools.com/python/python\_ml\_decision\_tree.asp\](https://www.w3schools.com/python/python_ml_decision_tree.asp/)

ANS=

# Import required libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

# Load the dataset

data = pd.read\_csv("shows.csv")

# Split the dataset into features (X) and target (y)

X = data[["Age", "Experience", "Rank", "Nationality"]]

y = data["Go"]

# Convert categorical data to numerical data

X = pd.get\_dummies(X)

# Create and train the Decision Tree Classifier

clf = DecisionTreeClassifier()

clf.fit(X, y)

# Define the new show details

new\_show = pd.DataFrame({

"Age": [40],

"Experience": [10],

"Rank": [7],

"Nationality": ["USA"]

})

# Convert the new show's features to match the model

new\_show = pd.get\_dummies(new\_show)

new\_show = new\_show.reindex(columns=X.columns, fill\_value=0)

# Predict the class label

prediction = clf.predict(new\_show)

print(f"Prediction for the new show: {prediction[0]}")

plt.figure(figsize=(12, 8))

plot\_tree(

clf,

feature\_names=["Age", "Experience", "Rank", "Nationality"],

class\_names=label\_encoder.classes\_,

filled=True,

rounded=True

)

plt.title("Decision Tree Visualization")

plt.show()

Prediction for the new show: Yes

SET B 1. Consider following dataset weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rain y','Sunny','Overcast','Overcast','Rainy'] temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild '] play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Y es','No']. Use Naïve Bayes algorithm to predict[ 0:Overcast, 2:Mild] tuple belongs to which class whether to play the sports or not.

Ans=

# Import required libraries

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.naive\_bayes import GaussianNB

# Dataset

weather = ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny',

'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy']

temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild',

'Mild', 'Mild', 'Hot', 'Mild']

play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

# Encode categorical features to numeric

le\_weather = LabelEncoder()

le\_temp = LabelEncoder()

le\_play = LabelEncoder()

weather\_encoded = le\_weather.fit\_transform(weather) # Encodes weather to numbers

temp\_encoded = le\_temp.fit\_transform(temp) # Encodes temp to numbers

play\_encoded = le\_play.fit\_transform(play) # Encodes play to numbers

# Combine features into a single dataset

features = np.array(list(zip(weather\_encoded, temp\_encoded)))

# Build and train the Naive Bayes model

model = GaussianNB()

model.fit(features, play\_encoded)

# Encode the input tuple [Overcast, Mild]

input\_data = np.array([[le\_weather.transform(['Overcast'])[0], le\_temp.transform(['Mild'])[0]]])

# Predict the class for the input

predicted = model.predict(input\_data)

predicted\_label = le\_play.inverse\_transform(predicted)

print(f"Prediction for [Overcast, Mild]: {predicted\_label[0]}")

output=

Prediction for [Overcast, Mild]: Yes