Name : Dion D Rodrigues Assignment.no: 04

Roll.no : 53 Class : FYMCA B Subject : ML Lab Batch : B3

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

# Importing all necessary libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import classification\_report

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

# Set display options to show all rows and columns

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

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

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

# Step 1: Importing the dataset

# Use the appropriate encoding to avoid UnicodeDecodeError

dataset = pd.read\_csv('C:/Users/Admin/Desktop/ML/Assignment\_4/spotify\_2023.csv', encoding='ISO-8859-1')

# Displaying the first few rows of the dataset to understand its structure

print(dataset)

# Step 2: Filtering out columns

# Assuming columns 2 and 3 contain features like age and salary, and column 4 contains labels

X = dataset.iloc[:, [2, 3]] # Features: age and salary (modify column indices as needed)

Y = dataset.iloc[:, 4] # Labels: Modify the index as per the dataset structure

# Step 3: Data split for training and testing (75% training, 25% testing)

# Splitting the dataset into training and testing sets

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=0)

# Step 4: Scaling the data

# Scaling features to ensure they follow a Normal Distribution

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Step 5: Building the model using the RBF kernel

# Initializing the Support Vector Classifier with an RBF kernel

classifier\_rbf = SVC(kernel='rbf', random\_state=0)

classifier\_rbf.fit(X\_train, Y\_train) # Training the model on the training data

# Step 6: Making predictions on the test data

Y\_pred\_rbf = classifier\_rbf.predict(X\_test)

# Step 7: Printing the confusion matrix

# A confusion matrix helps evaluate the accuracy of the classification

cm\_rbf = confusion\_matrix(Y\_test, Y\_pred\_rbf)

print("Confusion Matrix:\n", cm\_rbf)

# Step 8: Classification report

# Use zero\_division=0 to handle undefined precision when there are no predicted samples for a label

class\_report\_rbf = classification\_report(Y\_test, Y\_pred\_rbf, zero\_division=0)

print("Classification Report:\n", class\_report\_rbf)



