Assignment No.9

import numpy as np

from sklearn.datasets import load\_iris

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

from sklearn.svm import SVC

from sklearn import metrics

import matplotlib.pyplot as plt

# Load the Iris dataset

iris = load\_iris()

X = iris.data[iris.target != 0] # Selecting only two classes (1 and 2)

y = iris.target[iris.target != 0]

# Use only the first two features for simplicity

X = X[:, :2]

# 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)

# Create and train the SVM classifier with a linear kernel

svm\_classifier = SVC(kernel='linear', random\_state=42)

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

# Make predictions on the test set

y\_pred = svm\_classifier.predict(X\_test)

# Evaluate model performance

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

print("Accuracy:", accuracy)

# Visualizing the decision boundary (only for 2D data)

def plot\_decision\_boundary(X, y, model):

# Create a mesh grid for plotting decision boundary

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01),

np.arange(y\_min, y\_max, 0.01))

# Predict on the mesh grid points

Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# Plotting the decision boundary and training points

plt.contourf(xx, yy, Z, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('SVM Decision Boundary')

plt.show()

plot\_decision\_boundary(X\_train, y\_train, svm\_classifier)

#OUTPUT

Accuracy: 0.7

