import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.svm import SVC

from sklearn.metrics import plot\_confusion\_matrix

from sklearn.preprocessing import StandardScaler

# Load the breast cancer dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

column\_names = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness',

'Mean Compactness', 'Mean Concavity', 'Mean Concave Points', 'Mean Symmetry', 'Mean Fractal Dimension',

'SE Radius', 'SE Texture', 'SE Perimeter', 'SE Area', 'SE Smoothness', 'SE Compactness', 'SE Concavity',

'SE Concave Points', 'SE Symmetry', 'SE Fractal Dimension', 'Worst Radius', 'Worst Texture',

'Worst Perimeter', 'Worst Area', 'Worst Smoothness', 'Worst Compactness', 'Worst Concavity',

'Worst Concave Points', 'Worst Symmetry', 'Worst Fractal Dimension']

data = pd.read\_csv(url, header=None, names=column\_names)

# Drop unnecessary columns (e.g., ID column)

data = data.drop(['ID'], axis=1)

# Convert categorical diagnosis labels to numerical (Malignant: 1, Benign: 0)

data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})

# Separate features (X) and target variable (y)

X = data.drop('Diagnosis', axis=1)

y = data['Diagnosis']

# Consider only the first two features for simplicity

X = X.iloc[:, :2]

# 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)

# Standardize features (important for SVM)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Initialize the SVM classifier

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

# Train the classifier on the scaled training data

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

# Plot Decision Boundary

plt.figure(figsize=(10, 6))

plt.scatter(X\_train\_scaled[:, 0], X\_train\_scaled[:, 1], c=y\_train, cmap=plt.cm.Paired, edgecolors='k', s=30)

ax = plt.gca()

xlim = ax.get\_xlim()

ylim = ax.get\_ylim()

# Create grid to evaluate model

xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 100), np.linspace(ylim[0], ylim[1], 100))

Z = svm\_classifier.decision\_function(np.c\_[xx.ravel(), yy.ravel()])

# Plot decision boundary

Z = Z.reshape(xx.shape)

plt.contour(xx, yy, Z, colors='k', levels=[0], alpha=0.5, linestyles=['-'])

# Plot support vectors

plt.scatter(svm\_classifier.support\_vectors\_[:, 0], svm\_classifier.support\_vectors\_[:, 1], s=100, linewidth=1, facecolors='none', edgecolors='k')

plt.title('SVM Decision Boundary')

plt.xlabel('Feature 1 (Standardized)')

plt.ylabel('Feature 2 (Standardized)')

plt.show()

# Confusion Matrix Plot

disp = plot\_confusion\_matrix(svm\_classifier, X\_test\_scaled, y\_test, cmap=plt.cm.Blues, display\_labels=['Benign', 'Malignant'])

disp.ax\_.set\_title('Confusion Matrix')

plt.show()



