**Bahria University,**

Karachi Campus



## LAB EXPERIMENT NO.

## 7

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Write a python program to implement the support vector machine on Diabetes dataset. implement the following different kernels of SVM and compare the accuracy score and visualize the confusion matrix and hyperplane. |
| **2** | Design the workflow with the help of KNIME to implement the Support Vector Machine Algorithm on any classification dataset |

**Submitted On:**

18 feb 2024

(Date: DD/MM/YY)

**TASK 1: Write a python program to implement the support vector machine on Diabetes dataset. implement the following different kernels of SVM and compare the accuracy score and visualize the confusion matrix and hyperplane.**

**SOLUTION:**

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

import numpy as np

file\_path = 'diabetes.csv'

diabetes\_data = pd.read\_csv(file\_path)

diabetes\_data.head(), diabetes\_data.describe(), diabetes\_data.isnull().sum()

columns\_to\_fix = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for col in columns\_to\_fix:

diabetes\_data[col] = diabetes\_data[col].replace(0, diabetes\_data[col].median())

X = diabetes\_data.drop('Outcome', axis=1)

y = diabetes\_data['Outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

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

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

X\_train\_scaled[:5]

columns\_to\_fix = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for col in columns\_to\_fix:

diabetes\_data[col] = diabetes\_data[col].replace(0, diabetes\_data[col].median())

X = diabetes\_data.drop('Outcome', axis=1)

y = diabetes\_data['Outcome']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

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

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

kernels = ['linear', 'poly', 'rbf']

models = {}

predictions = {}

accuracy\_scores = {}

for kernel in kernels:

svm\_model = SVC(kernel=kernel, gamma='auto')

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

y\_pred = svm\_model.predict(X\_test\_scaled)

models[kernel] = svm\_model

predictions[kernel] = y\_pred

accuracy\_scores[kernel] = accuracy\_score(y\_test, y\_pred)

print(accuracy\_scores)

fig, axes = plt.subplots(1, 3, figsize=(18, 5))

fig.suptitle('Confusion Matrices for Different SVM Kernels')

for ax, kernel in zip(axes.flatten(), kernels):

confusion\_mtx = confusion\_matrix(y\_test, predictions[kernel])

sns.heatmap(confusion\_mtx, annot=True, fmt='d', cmap='Blues', ax=ax)

ax.set\_xlabel('Predicted labels')

ax.set\_ylabel('True labels')

ax.set\_title(f'Kernel: {kernel}')

plt.tight\_layout(rect=[0, 0.03, 1, 0.95])

plt.show()

def plot\_hyperplane(clf, X, y, h=0.02, draw\_sv=True, title="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, h),

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

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

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

Z = Z.reshape(xx.shape)

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

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')

if draw\_sv:

sv = clf.support\_vectors\_

plt.scatter(sv[:, 0], sv[:, 1], c='yellow', marker='x', s=100, linewidths=2, edgecolors='k')

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

Z = Z.reshape(xx.shape)

plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])

plt.title(title)

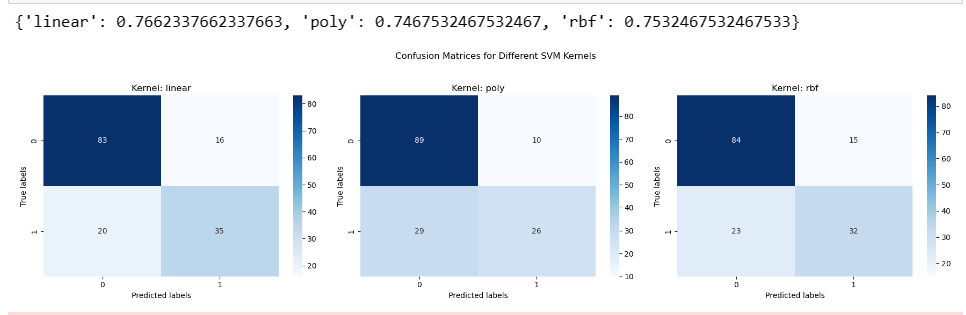
plt.xlabel('Principal Component 1')

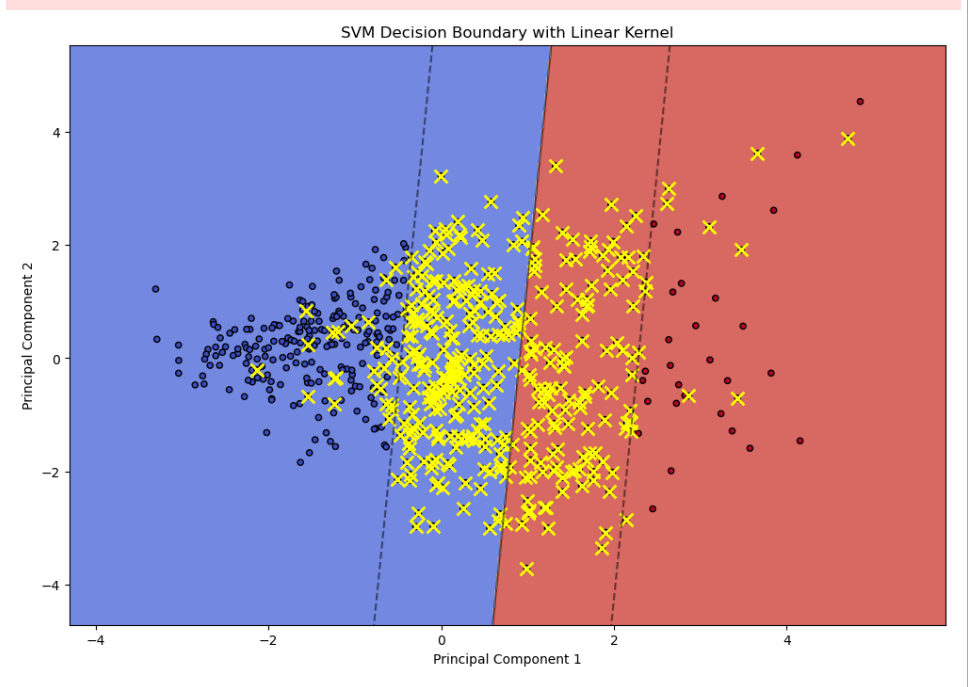
plt.ylabel('Principal Component 2')

plt.show()

plot\_hyperplane(svm\_model\_pca, X\_train\_pca, y\_train, title="SVM Decision Boundary with Linear Kernel")

**OUTPUT:**

****

****

**TASK 2:** **Design the workflow with the help of KNIME to implement the Support Vector Machine Algorithm on any classification dataset**

