**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.

import pandas as pd

from sklearn.svm import SVC

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

from sklearn.metrics import accuracy\_score, confusion\_matrix

import seaborn as sns

def plot\_decision\_boundary(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()

file\_path = 'diabetes.csv'

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

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']

scaler = StandardScaler()

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

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

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

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

models = {}

predictions = {}

accuracy\_scores = {}

for kernel in kernels:

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

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

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

    models[kernel] = svm\_model

    predictions[kernel] = y\_pred

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

print("Accuracy Scores:")

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='pink', ax=ax)

    ax.set\_xlabel('Predicted labels')

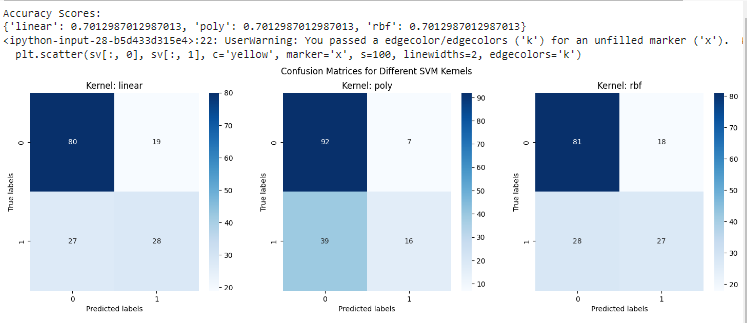
    ax.set\_ylabel('True labels')

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

selected\_kernel = 'poly'

svm\_model\_selected = models[selected\_kernel]

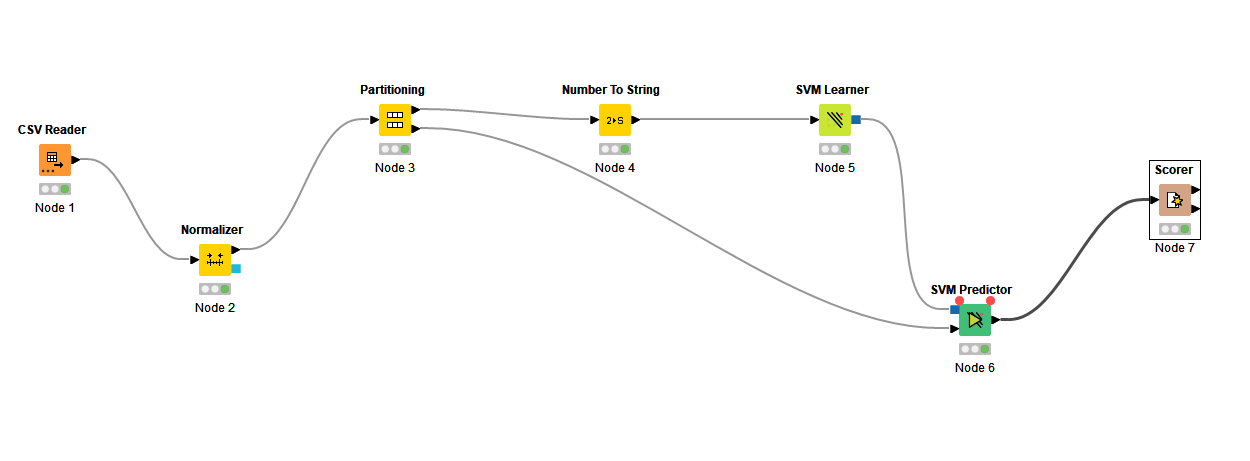
plot\_decision\_boundary(svm\_model\_selected, X\_pca, y, title=f"SVM Decision Boundary with {selected\_kernel.capitalize()} Kernel")



A blue and red background with yellow dots

Description automatically generated

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



A screenshot of a computer

Description automatically generated