Bahria University,

Karachi Campus

A logo with text on it

Description automatically generated

LAB EXPERIMENT NO.

07

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **01** | 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. |
| **02** | Design the workflow with the help of KNIME to implement the Support Vector Machine Algorithm on any classification dataset |
|  |  |

Submitted On:

03-04-2024

(Date: DD/MM/YYYY)

**Task No. 01**: 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

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.svm import SVC

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

from sklearn.preprocessing import StandardScaler

data = pd.read\_csv("BooZSY.csv")

data

# Split the data into features (X) and target variable (y)

X = data.drop("Outcome", axis=1)

y = data['Outcome']

# 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)

# Scale the features

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

# Create an SVM model instance

svm\_model = SVC (kernel='linear', C=1.0, random\_state=42)

# Train the SVM model

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

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

# Make predictions on the test set y\_pred = svm\_model.predict(x\_test)

#Evaluate the model's performance

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

print (f'Accuracy: {accuracy}')

# Generate and print classification report and confusion matrix

print("Classification Report:")

print(classification\_report (y\_test, y\_pred))

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

**OUTPUT:**

A screenshot of a computer

Description automatically generated

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

h = .02 # step size in the mesh

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))

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

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8, colors='green') # Changing color to 'viridis'

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

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Decision Boundary')

plt.show()

# Assuming X\_train has 2 features for visualization

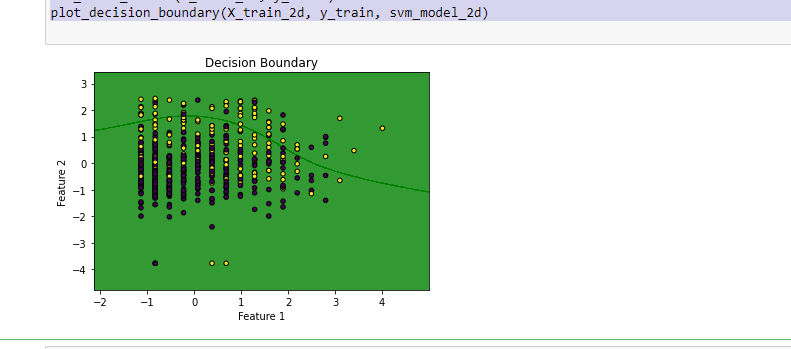
X\_train\_2d = x\_train[:, :2]

svm\_model\_2d = SVC(kernel='poly', C=1.0, random\_state=42)

svm\_model\_2d.fit(X\_train\_2d, y\_train)

plot\_decision\_boundary(X\_train\_2d, y\_train, svm\_model\_2d)

**OUTPUT:**

import

matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, classification\_report

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Greys') # Change 'Oranges' to your desired color palette

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.show()

**OUTPUT:**

A graph with a black and white squares

Description automatically generated with medium confidence

**Task No. 02**: Design the workflow with the help of KNIME to implement the Support Vector

Machine Algorithm on any classification dataset.

**Solution:**

A diagram of a computer

Description automatically generated

**Output:**

