Name : Dion D Rodrigues Assignment.no: 06

Roll.no : 53 Class : FYMCA B Subject : ML Lab Batch : B3

Here’s a step-by-step format of your **K-Nearest Neighbors (KNN) classifier** program to help create a well-structured Word document. You can copy and paste this into a Word file for easy reading and formatting.

**Step-by-Step Implementation of KNN Classifier**

**Step 1: Importing Libraries**

First, import the necessary libraries to handle data, create the model, and evaluate its performance.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

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

* **Pandas**: For handling and manipulating data.
* **NumPy**: For numerical operations.
* **Matplotlib**: For plotting the decision boundary.
* **Sklearn**: For model creation, data splitting, and evaluation.

Here’s a step-by-step format of your **K-Nearest Neighbors (KNN) classifier** program to help create a well-structured Word document. You can copy and paste this into a Word file for easy reading and formatting.

**Step-by-Step Implementation of KNN Classifier**

**Step 1: Importing Libraries**

First, import the necessary libraries to handle data, create the model, and evaluate its performance.

python

Copy code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

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

* **Pandas**: For handling and manipulating data.
* **NumPy**: For numerical operations.
* **Matplotlib**: For plotting the decision boundary.
* **Sklearn**: For model creation, data splitting, and evaluation.

**Step 2: Loading the Dataset**

df = pd.read\_csv('student\_performance.csv')

**Step 3: Defining Features and Target**

Select the relevant columns (Hours\_Studied, Hours\_Slept) as features (input), and Passed as the target (output).

X = df[['Hours\_Studied', 'Hours\_Slept']]

y = df['Passed']

**Step 4: Splitting the Dataset**

Split the dataset into **training** and **test sets**. Here, 80% of the data is used for training the model, while 20% is used for testing.

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

**Step 5: Initializing the KNN Classifier**

Create a **K-Nearest Neighbors** classifier and set the number of neighbors (**K=3** in this case).

knn = KNeighborsClassifier(n\_neighbors=3)

**Step 6: Training the KNN Model**

Train the KNN model on the training dataset using the fit() function.

knn.fit(X\_train, y\_train)

**Step 7: Making Predictions**

Use the trained model to predict the outcomes for the test set.

y\_pred = knn.predict(X\_test)

**Step 8: Evaluating the Model**

Evaluate the model's performance by calculating **accuracy** and generating a **classification report**.

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

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")

print(report)

**Step 9: Plotting the Decision Boundary**

Visualize the decision boundary that the KNN classifier creates based on the features Hours\_Studied and Hours\_Slept. Use a mesh grid to plot the boundary and display the training and test data points.

# Create a mesh grid

h = 0.2 # step size in the mesh

x\_min, x\_max = X['Hours\_Studied'].min() - 1, X['Hours\_Studied'].max() + 1

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

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

# Create a DataFrame for the mesh grid with the same feature names as the training data

grid\_points = pd.DataFrame(np.c\_[xx.ravel(), yy.ravel()], columns=X.columns)

# Predict on the mesh grid

Z = knn.predict(grid\_points)

Z = Z.reshape(xx.shape)

# Plot the contours

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

plt.contourf(xx, yy, Z, alpha=0.5, cmap=plt.cm.RdYlBu)

plt.scatter(X\_train['Hours\_Studied'], X\_train['Hours\_Slept'], c=y\_train, marker='o', edgecolor='k', label='Training data')

plt.scatter(X\_test['Hours\_Studied'], X\_test['Hours\_Slept'], c=y\_test, marker='x', label='Test data') # Removed edgecolor='k'

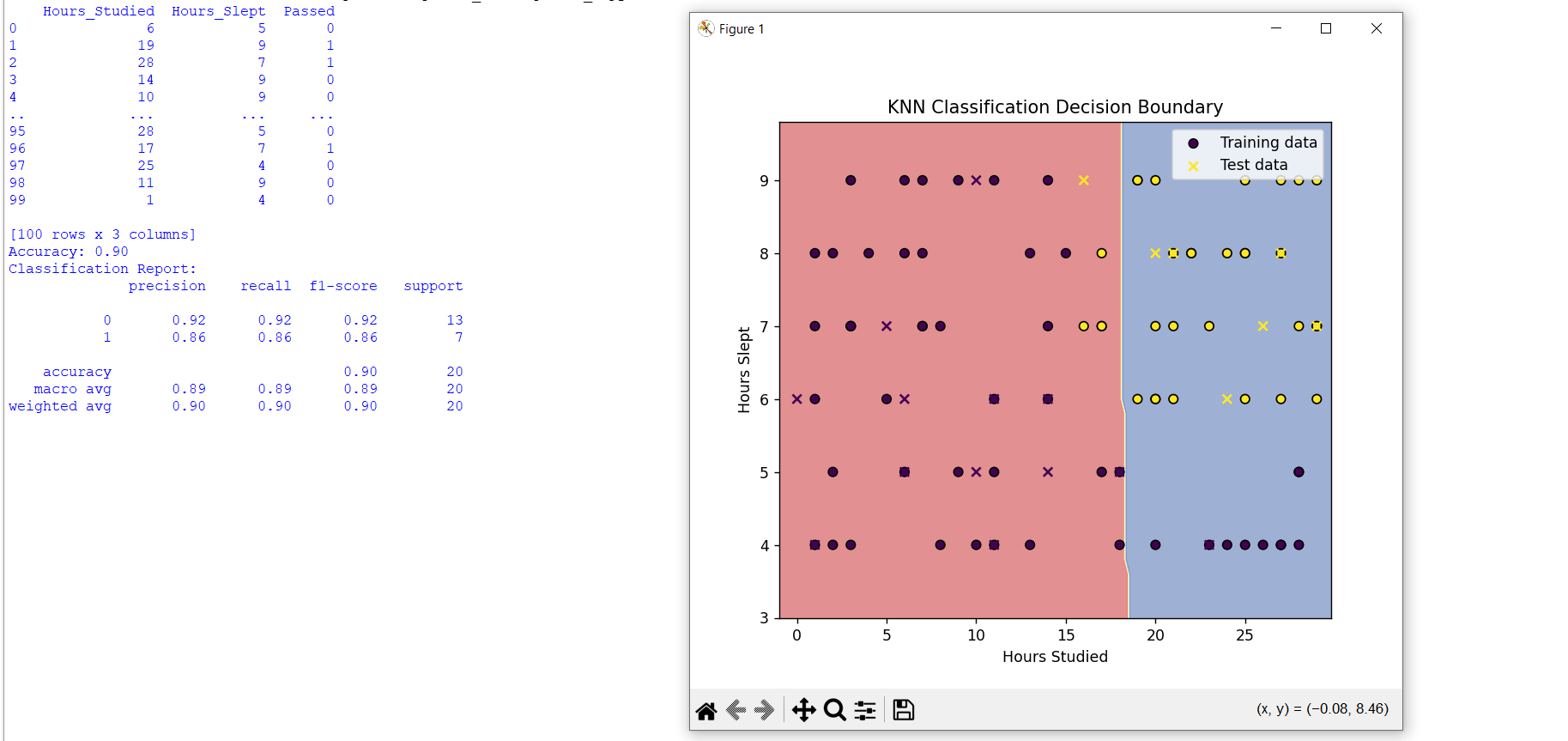
plt.xlabel('Hours Studied')

plt.ylabel('Hours Slept')

plt.title('KNN Classification Decision Boundary')

plt.legend()

plt.show()

**Output:**