LAB ASSIGNMENT -4

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.metrics import confusion\_matrix, classification\_report, mean\_squared\_error, r2\_score

from sklearn.neighbors import KNeighborsClassifier

# Load the dataset

df = pd.read\_csv('diabetes.csv')

# Split the dataset into features and target variable

X = df.drop('Outcome', axis=1)

y = df['Outcome']

# Select two features for visualization and classification (A6)

feature\_1 = 'Glucose'

feature\_2 = 'BMI'

X\_selected = df[[feature\_1, feature\_2]]

# Split into training and test sets

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

# Train and classify using kNN (k=3)

kNN\_model = KNeighborsClassifier(n\_neighbors=3)

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

y\_pred\_train = kNN\_model.predict(X\_train)

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

# A1: Evaluate confusion matrix and classification report

print("Training Data:")

print("Confusion Matrix:\n", confusion\_matrix(y\_train, y\_pred\_train))

print("Classification Report:\n", classification\_report(y\_train, y\_pred\_train))

print("Test Data:")

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_test))

print("Classification Report:\n", classification\_report(y\_test, y\_pred\_test))

# A2: Calculate error metrics (MSE, RMSE, MAPE, R2)

mse = mean\_squared\_error(y\_test, y\_pred\_test)

rmse = np.sqrt(mse)

mape = np.mean(np.abs((y\_test - y\_pred\_test) / y\_test)) \* 100

r2 = r2\_score(y\_test, y\_pred\_test)

print(f"MSE: {mse}, RMSE: {rmse}, MAPE: {mape}%, R2: {r2}")

# A3: Scatter plot for training data (using selected features)

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

plt.scatter(X\_train[y\_train == 0][feature\_1], X\_train[y\_train == 0][feature\_2], color='blue', label='Class 0')

plt.scatter(X\_train[y\_train == 1][feature\_1], X\_train[y\_train == 1][feature\_2], color='red', label='Class 1')

plt.xlabel(feature\_1)

plt.ylabel(feature\_2)

plt.legend()

plt.title('Training Data (Selected Features)')

plt.show()

# A4 & A5: Scatter plot for test data classification results

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

plt.scatter(X\_test[y\_pred\_test == 0][feature\_1], X\_test[y\_pred\_test == 0][feature\_2], color='blue', alpha=0.5, label='Class 0')

plt.scatter(X\_test[y\_pred\_test == 1][feature\_1], X\_test[y\_pred\_test == 1][feature\_2], color='red', alpha=0.5, label='Class 1')

plt.xlabel(feature\_1)

plt.ylabel(feature\_2)

plt.legend()

plt.title('Test Data Classification (k=3)')

plt.show()

# A7: Hyperparameter tuning using GridSearchCV

param\_grid = {'n\_neighbors': range(1, 21)}

grid\_search = GridSearchCV(KNeighborsClassifier(), param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

print(f"Best k: {grid\_search.best\_params\_['n\_neighbors']}")

Output:

Training Data:

Confusion Matrix:

[[364 37]

[ 62 151]]

Classification Report:

precision recall f1-score support

0 0.85 0.91 0.88 401

1 0.80 0.71 0.75 213

accuracy 0.84 614

macro avg 0.83 0.81 0.82 614

weighted avg 0.84 0.84 0.84 614

Test Data:

Confusion Matrix:

[[81 18]

[26 29]]

Classification Report:

precision recall f1-score support

0 0.76 0.82 0.79 99

1 0.62 0.53 0.57 55

accuracy 0.71 154

macro avg 0.69 0.67 0.68 154

weighted avg 0.71 0.71 0.71 154

MSE: 0.2857142857142857, RMSE: 0.5345224838248488, MAPE: inf%, R2: -0.2444444444444449

A diagram of red and blue dots

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.