**Lab-4**

CHANDRA KIRAN KOPPARAPU

BL.EN. U4AIE23141

Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

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

from sklearn.neighbors import KNeighborsClassifier

# Load dataset

df = pd.read\_excel("Dataset.xlsx", sheet\_name="Dataset")

# Selecting features and target variable

features = ['person\_age', 'person\_income']  # Selecting two numerical features for simplicity

target = 'loan\_status'

X = df[features]

y = df[target]

# Splitting data

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

# Scaling the data

scaler = StandardScaler()

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

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

# Train kNN model (k=3)

knn = KNeighborsClassifier(n\_neighbors=3)

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

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

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

# A1: Confusion Matrix and Performance Metrics

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

print("Confusion Matrix:\n", conf\_matrix)

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

# A2: Regression Metrics (assuming a previous price prediction task)

y\_actual = np.random.rand(len(y\_test)) \* 100  # Placeholder values

y\_pred = y\_actual + np.random.normal(0, 10, len(y\_test))

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

rmse = np.sqrt(mse)

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

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

print("MSE:", mse)

print("RMSE:", rmse)

print("MAPE:", mape)

print("R2 Score:", r2)

# A3: Generate and visualize training data for kNN

np.random.seed(42)

X\_train\_knn = np.random.uniform(1, 10, (20, 2))

y\_train\_knn = np.random.choice([0, 1], 20)

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

plt.scatter(X\_train\_knn[:, 0], X\_train\_knn[:, 1], c=y\_train\_knn, cmap='bwr', edgecolors='k')

plt.xlabel("Feature X")

plt.ylabel("Feature Y")

plt.title("Training Data")

plt.show()

# A4: Generate and classify test data using kNN

test\_points = np.array([[x, y] for x in np.arange(0, 10, 0.1) for y in np.arange(0, 10, 0.1)])

preds = knn.predict(test\_points)

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

plt.scatter(test\_points[:, 0], test\_points[:, 1], c=preds, cmap='bwr', alpha=0.5, edgecolors='k')

plt.xlabel("Feature X")

plt.ylabel("Feature Y")

plt.title("Test Data Classification")

plt.show()

# A5: Repeat A4 for multiple k values

for k in [1, 5, 10]:

    knn\_k = KNeighborsClassifier(n\_neighbors=k)

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

    preds\_k = knn\_k.predict(test\_points)

    plt.figure(figsize=(8, 6))

    plt.scatter(test\_points[:, 0], test\_points[:, 1], c=preds\_k, cmap='bwr', alpha=0.5, edgecolors='k')

    plt.xlabel("Feature X")

    plt.ylabel("Feature Y")

    plt.title(f"Test Data Classification (k={k})")

    plt.show()

# A6: Repeat for real project data

knn\_real = KNeighborsClassifier(n\_neighbors=3)

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

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

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

# A7: Hyperparameter tuning

param\_grid = {'n\_neighbors': range(1, 20)}

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

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

print("Best k value:", grid\_search.best\_params\_['n\_neighbors'])