R.V.Raghunandan

AIE23147

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

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

def generate\_training\_data():

np.random.seed(42)

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

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

return X\_train, y\_train

def generate\_test\_data():

x\_test = np.arange(0, 10, 0.1)

y\_test = np.arange(0, 10, 0.1)

X\_test = np.array(np.meshgrid(x\_test, y\_test)).T.reshape(-1, 2)

return X\_test

def train\_knn(X\_train, y\_train, k=3):

knn = KNeighborsClassifier(n\_neighbors=k)

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

return knn

def evaluate\_model(knn, X\_train, y\_train, X\_test, y\_test):

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

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

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

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

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

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

def tune\_hyperparameter(X\_train, y\_train):

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

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

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

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

return grid\_search.best\_params\_['n\_neighbors']

def plot\_training\_data(X\_train, y\_train):

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

plt.title("Training Data")

plt.xlabel("X Feature")

plt.ylabel("Y Feature")

plt.show()

def plot\_test\_results(X\_test, y\_pred):

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_pred, cmap='bwr', s=1)

plt.title("Test Data Classification")

plt.xlabel("X Feature")

plt.ylabel("Y Feature")

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

X\_train, y\_train = generate\_training\_data()

X\_test = generate\_test\_data()

plot\_training\_data(X\_train, y\_train)

best\_k = tune\_hyperparameter(X\_train, y\_train)

knn\_model = train\_knn(X\_train, y\_train, best\_k)

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

plot\_test\_results(X\_test, y\_pred\_test)