Cse23254lab 4

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

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

data = pd.read\_csv("features\_raw (1).csv")

X = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

# A1

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

knn3 = KNeighborsClassifier(n\_neighbors=3)

knn3.fit(X\_train, y\_train)

y\_pred\_train = knn3.predict(X\_train)

y\_pred\_test = knn3.predict(X\_test)

cm\_train = confusion\_matrix(y\_train, y\_pred\_train)

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

report\_train = classification\_report(y\_train, y\_pred\_train, output\_dict=True)

report\_test = classification\_report(y\_test, y\_pred\_test, output\_dict=True)

# A2 (example: assume first column is feature, last column numeric target for regression)

y\_reg = data.iloc[:, -1].values.astype(float)

X\_reg = data.iloc[:, :-1].values

Xr\_train, Xr\_test, yr\_train, yr\_test = train\_test\_split(X\_reg, y\_reg, test\_size=0.3)

from sklearn.linear\_model import LinearRegression

reg = LinearRegression()

reg.fit(Xr\_train, yr\_train)

yr\_pred = reg.predict(Xr\_test)

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

rmse = np.sqrt(mse)

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

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

# A3

np.random.seed(0)

X\_train2 = np.random.randint(1, 11, (20, 2))

y\_train2 = np.random.randint(0, 2, 20)

colors = ['blue' if c == 0 else 'red' for c in y\_train2]

plt.scatter(X\_train2[:, 0], X\_train2[:, 1], c=colors)

plt.show()

# A4

x1 = np.arange(0, 10.1, 0.1)

x2 = np.arange(0, 10.1, 0.1)

xx1, xx2 = np.meshgrid(x1, x2)

X\_test2 = np.c\_[xx1.ravel(), xx2.ravel()]

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

knn\_demo.fit(X\_train2, y\_train2)

y\_pred2 = knn\_demo.predict(X\_test2)

colors2 = ['blue' if c == 0 else 'red' for c in y\_pred2]

plt.scatter(X\_test2[:, 0], X\_test2[:, 1], c=colors2, s=1)

plt.show()

# A5

for k in [1, 3, 5, 7, 9]:

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

knn\_demo.fit(X\_train2, y\_train2)

y\_pred2 = knn\_demo.predict(X\_test2)

colors2 = ['blue' if c == 0 else 'red' for c in y\_pred2]

plt.scatter(X\_test2[:, 0], X\_test2[:, 1], c=colors2, s=1)

plt.title(f"k={k}")

plt.show()

# A6

X\_proj = X[:, :2]

y\_proj = y

X\_trainp, X\_testp, y\_trainp, y\_testp = train\_test\_split(X\_proj, y\_proj, test\_size=0.3)

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

knn\_proj.fit(X\_trainp, y\_trainp)

y\_predp = knn\_proj.predict(X\_testp)

plt.scatter(X\_proj[:, 0], X\_proj[:, 1], c=['blue' if c == list(np.unique(y))[0] else 'red' for c in y\_proj])

plt.show()

# A7

param\_grid = {'n\_neighbors': np.arange(1, 20)}

grid = GridSea