import numpy as np

import pandas as pd

# 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pd.options.display.max\_columns=None

pd.options.display.max\_rows=None

wheat\_df = pd.read\_csv('wheat.csv')

print(wheat\_df)

print(wheat\_df.info())

# 2 There is no categorical data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

X = wheat\_df.drop(columns=['Variety'])

y = wheat\_df['Variety']

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

# 5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

print("\n\nMean", np.mean(X\_train))

print("Standard Deviation", np.std(X\_train))

# 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.neighbors import KNeighborsClassifier

kNN\_model = KNeighborsClassifier(n\_neighbors = 8)

# 8, 10, 15

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

# 7 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

num\_correct = np.where(y\_predicts == y\_test, 1, 0).sum()

print('\n\nNumber correct is ', num\_correct, ' out of ', len(y\_predicts))

rate = num\_correct / len(y\_predicts) \* 100

print('which is around ', str(round(rate)) + '%')

print('\n\n \*\*\* SKLEARN ACCURACY INFORMATION \*\*\* ')

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_test, y\_predicts))

print('\n\nClassification report')

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,y\_predicts))

print('\n\nConfusion matrix')

from sklearn.metrics import confusion\_matrix

print(confusion\_matrix(y\_test,y\_predicts))

# 8 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.model\_selection import cross\_val\_score

k\_values = [i for i in range(1,50)]

scores = []

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors = k)

score = cross\_val\_score(knn,X,y,cv=4) # 4 train and 1 test from original data

scores.append(np.mean(score))

import seaborn as sns

import matplotlib.pyplot as plt

sns.lineplot(x=k\_values, y = scores,marker='o')

plt.xlabel("K values")

plt.ylabel("Accuracy score")

plt.show()

# 11 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

xpredict = [[19.09, 16.61, .8722, 6.3, 3.737, 6.682, 6.053]]

X\_predicts = pd.DataFrame(xpredict,columns=['Area','Perim','Compactness', 'Length', 'Width', 'AsymmCoeff', 'GrooveLength'])

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

y\_p = kNN\_model.predict(X\_predicts)

print('Recommendation is', y\_p)

xpredict = [[11.34, 12.82, .8596, 5.053, 2.9, 3.347, 4.999]]

X\_predicts = pd.DataFrame(xpredict,columns=['Area','Perim','Compactness', 'Length', 'Width', 'AsymmCoeff', 'GrooveLength'])

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

y\_p = kNN\_model.predict(X\_predicts)

print('Recommendation is', y\_p)