import numpy as np

import pandas as pd

def euclidean\_distance(x1,x2):

return np.sqrt(np.sum((x1-x2)\*\*2))

class KNN:

def \_\_init\_\_(self,k=3):

self.k=k

def fit(self,x,y):

self.x=x

self.y=y

def predict\_single(self, point):

# Calculate distances between x and all points in the training set

distances = []

for data\_point in self.x:

distance = euclidean\_distance(point, data\_point)

distances.append(distance)

# Sort distances in ascending order and get corresponding indices

sorted\_indices = np.argsort(distances)

# Select the indices of the k nearest neighbors

k\_nearest\_indices = sorted\_indices[:self.k]

# Extract labels of k nearest neighbors

k\_nearest\_labels = []

for index in k\_nearest\_indices:

label = self.y[index]

k\_nearest\_labels.append(label)

# Predict the label based on the majority class among the k nearest neighbors

predicted\_label = np.argmax(np.bincount(k\_nearest\_labels))

return predicted\_label

def predict(self, x):

# Generate predictions for multiple data points in the test set

predicted\_labels = []

for point in x:

predicted\_labels.append(self.predict\_single(point))

# Convert the list of predicted labels into a numpy array

# predicted\_labels\_array = np.array(predicted\_labels)

# Return the array of predicted labels

return (predicted\_labels)

df = pd.read\_csv(

'/Users/swastikagarwal/Downloads/SEM5\_RVCE/LAB WORK/AIML/iris\_csv (1).csv')

df.head()

x=df.iloc[:,:-1].values

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

# x

d={'Iris-versicolor':1, 'Iris-virginica':2, 'Iris-setosa':3}

for i in range(len(y)):

y[i]=d[y[i]]

print(set(y))

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=7)

model=KNN()

model.fit(x\_train,y\_train)

y\_pred=model.predict(x\_test)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

print("accuracy: ", accuracy\_score(list(y\_test), list(y\_pred)))

print("precision: ", precision\_score(list(y\_test), list(y\_pred), average='macro'))

print("recall: ", recall\_score(list(y\_test), list(y\_pred), average='macro'))

print("f1 score: ", f1\_score(list(y\_test), list(y\_pred), average='macro'))

valid = df.sample(n=20)

x\_valid = (valid.iloc[:, :-1].values)

y\_valid = (valid.iloc[:, -1].values)

y\_predict = (model.predict(x\_valid))

(y\_predict), (y\_valid)

print(accuracy\_score(list(y\_predict), list(y\_valid)))