Pseudo Code for KNN Classification

##understand the problem statement for wine quality Dataset

##import all necessary libraries

import pandas as pd

import numpy as np

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

##Load the wine quality dataset into as pandas dataframe

Data=pd.read\_csv("winequality.csv")

## Normalize the dataset by subtracting the mean and dividing by the standard deviation

Data\_normalized=(Data-mean)/std

##preprocess the data [eg:impute missing values,checking outliers..,]

##Exploratory Data Analysis(EDA)[eg:To check any outliers,missing values]and uses various visualization tools [eg:histograms, box plots,scatter plots]

##split the dataset into training and testing datasets

X\_split=(Data\_normalized.drop['quality'],axis=1)

y\_split=Data\_normalized['quality']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(Data\_normalized.drop['quality'],axis=1),Data\_normalized['quality'],test\_size=0.2,random\_state=34)

##calculate distance matrix

def euclidean\_distance(X\_train,X\_test):

for i in range(len(X\_test)):

return np.sqrt(np.sum(X\_test-X\_train)\*\*2))

##select the k-nearest neighbors of the instance based on the calculated distances

X=X\_split[2] # A datapoint to want to find the k-nearest neighbors

def get\_nearest\_neighbors(X\_split,X,K):

euclidean\_distance = []

for i in range(len(X\_train)):

d = np.sqrt(np.sum((X\_test-X\_train[i]\*\*2))

euclidean\_distance.append(i,d))

euclidean\_distance.sort(key=lambda X:X[1])

Neighbors=[distances[i][0] for i in range(k)]

return neighbors

##Classify the instance based on the majority class of the K-nearest neighbors.

def predict\_data(X\_test,X\_train,y\_train,K):

neighbors =get\_nearest\_neighbors(X\_split,X,K)

classes= [y\_train[i] for i in neighbors]

class\_count={}

for c in classes:

if c in class\_count:

class\_count[c]+=1

else :

class\_count[c]=1

prediction = Max(class\_count,key=class\_count.get)

return prediction

##Evaluate the classification accuracy on the testing dataset

##Repeat steps 4-8 for different values of K and distance metrics to find the best hyperparameters

##Output the best hyperparameters and the classification accuracy on the testing dataset