

CS 7267

MACHINE LEARNING

PROJECT 2

UNSUPERVISED LEARNING

#### INSTRUCTOR

**Dr. Mahmut KARAKAYA**

**Issouf kindo**

**I. ABSTRACT**

In this project implements the KNN Classification algorithm. Its takes row data in csv format and cluster the objects of the data set based of the Euclidian distance. The Project was coded in python, and it requires the pandas’, NumPy and sklearn package for data manipulation.

1. The dataset used in the testing process needed be normalized since the range of the first column is very high compared to the range of the last column. We used z-core normalization in this project.

**II. Results:**

Text

Description automatically generated

**Text

Description automatically generated**

III. Discussion/Conclusion

Varies the K value from 1 to 9 have shown that k =7 give the highest accuracy of .978. In addition, since this is medical dataset, the FN negative rate must be the lowest, which k = 7 has the minimum value of 3. Overall lessons learned in this project, are implementing the KNN algorithm and selecting K for better result. In addition, the project teaches the implementation of important evaluation metrics such as accuracy and confusion matrix.

* 1. **source code:**
* import pandas as pd  
  import numpy as np  
  from collections import Counter  
  from sklearn.model\_selection import train\_test\_split  
    
    
  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\_train = X  
   self.y\_train = y  
    
   def predict(self, X):  
   predicted\_labels = [self.\_predict(x) for x in X]  
   return predicted\_labels  
    
   def \_predict(self, x):  
   distances = [euclidean\_distance(x, x\_train) for x\_train in self.X\_train]  
   k\_indices = np.argsort(distances)[:self.k]  
   k\_labels = [self.y\_train[i] for i in k\_indices]  
    
   # get the most common element in the k neighbors  
   most\_common = Counter(k\_labels).most\_common(1)  
   return most\_common[0][0]  
    
    
  # apply the z-score method in Pandas using the .mean() and .std() methods  
  def z\_score(df):  
   # copy the dataframe  
   df\_std = df.copy()  
   # apply the z-score method  
   for column in df\_std.columns:  
   df\_std[column] = (df\_std[column] - df\_std[column].mean()) / df\_std[column].std()  
    
   return df\_std  
    
    
  # Press the green button in the gutter to run the script.  
  if \_\_name\_\_ == '\_\_main\_\_':  
   df = pd.read\_csv("wdbc.data.mb.csv", names=[x for x in range(31)])  
   data = z\_score(df.loc[:, :29])  
   X, y = data.loc[:, :29], df.loc[:, 30]  
   X\_train, X\_test, y\_train, y\_test = train\_test\_split(X.to\_numpy(), y.to\_numpy(), test\_size=.3, random\_state=1234)  
    
   for k in [1, 3, 5, 7, 9]:  
   clf = KNN(k=k)  
   clf.fit(X\_train, y\_train)  
   pred = clf.predict(X\_test)  
    
   confusion\_matrix = {"TP": 0, "FN": 0, "TN": 0, "FP": 0}  
   for index, value in enumerate(pred):  
   if value == y\_test[index] and value == 1:  
   confusion\_matrix["TP"] += 1  
   if value == y\_test[index] and value == -1:  
   confusion\_matrix["TN"] += 1  
   if value != y\_test[index] and value == 1:  
   confusion\_matrix["FP"] += 1  
   if value != y\_test[index] and value == -1:  
   confusion\_matrix["FN"] += 1  
   print("--------confusion matrix for k = ",k,"-----------------")  
   tm = "\t +1\t\t -1\n +1\t TP: " + str(confusion\_matrix["TP"]) + "\t FN: " + str(  
   confusion\_matrix["FN"]) + "\n -1\t FP: " + str(confusion\_matrix["FN"]) + "\t TN: " + str(  
   confusion\_matrix["TN"])  
    
   print(tm)  
   accuracy = np.sum(pred == y\_test) / len(y\_test)  
   print("\n Accuracy: ",accuracy)  
   print("\_+\_+\_+\_+\_+\_+\_+\_+End of k = ",k,"\_+\_+\_+\_+\_+\_+\_+\_+\_+\_+\_+\_+\_+\n")