Thomas Freeman

4/14/22

Bioinformatics for CS

Report for Hw6

**Abstract:**

The goal of this assignment was to implement the KNN or K nearest neighbors’ algorithm on two different datasets. One we were provided with 4 features total, with the algorithm running for all four features. The other being the Iris dataset that we were given in Homework five, explicitly stating that KNN should analyze only the petal and sepal lengths. We also only needed to use three distinct K values, 3,5, and 7.

**Method:**

To implement the KNN algorithm we need to make sure the following is done. K nearest neighbors allows us to find distance between datapoints, based upon a given input for the number of distances to find, being K. For example with K = 7, we are finding the seven closest points To a data point, effectively performing analysis on a dataset in real time. I’ve also gone and calculated the mean error for k values one through eight, To make it easier to see the trend between K values 3,5, and 7.

**Discussion:**

With continued use of the algorithm and training on the dataset using different training sets, we could reduce the effects that K values have as each value depending on the input will produce a certain amount of error. Further training has potential to reduce the error, but that would require methods more complex and productive like Simple Vector machines (SVM). Subtracting the error rate value from 1 will give the true value of K from the graph. On average after multiple rounds of testing, our given dataset gives around 60% to 70% accuracy for the given K values, while the Iris dataset produces around 90% accuracy, with the lowest being 80%. This would show that our given dataset generates much more error then the Iris dataset as what we were given doesn’t have features that correspond to actual measurements. Iris has data that correspond to real measurements; therefore, it makes sense to be more accurate.

**Results:**

Here are the following graphs for the corresponding k values in each dataset. As stated in the discussion, subtracting the mean error from one would yield the true K value. This is shown in the classification report of the program along with a confusion matrix, to show True positives, False positives, True Negatives, and False negatives. This is more prevalent with our given dataset as there would be between 40 and 50 True Negatives and 10 to 20 false negatives. This indicates that not all features are reported to complete accuracy, in most datasets complete precision is nearly impossible.

First are the values when K = 3, 5 and 7 for the HW6 dataset

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Then we have the graphs for K = 3,5,7 for the Iris dataset

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

**Appendix**

The following code is used for both the HW6 and Iris dataset.

In [ ]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.datasets import load\_iris

In [ ]:

df = pd.read\_csv('Iris.csv')

df

Out[ ]:

|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 6 columns

In [ ]:

X = df.iloc[:,1 :3].values

y = df.iloc[:, 5].values

In [ ]:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

In [ ]:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

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

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

In [ ]:

from sklearn.neighbors import KNeighborsClassifier

# n\_neighbors is k

classifier = KNeighborsClassifier(n\_neighbors=5)

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

Out[ ]:

KNeighborsClassifier()

In [ ]:

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

In [ ]:

from sklearn.metrics import classification\_report, confusion\_matrix,accuracy\_score

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

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

Accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", Accuracy)

[[12 1 0]

[ 0 7 1]

[ 0 4 5]]

precision recall f1-score support

Iris-setosa 1.00 0.92 0.96 13

Iris-versicolor 0.58 0.88 0.70 8

Iris-virginica 0.83 0.56 0.67 9

accuracy 0.80 30

macro avg 0.81 0.78 0.78 30

weighted avg 0.84 0.80 0.80 30

Accuracy: 0.8

In [ ]:

error = []

# Calculating error for K values between 1 and 8

for i in range(1, 8):

knn = KNeighborsClassifier(n\_neighbors=i)

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

pred\_i = knn.predict(X\_test)

error.append(np.mean(pred\_i != y\_test))

In [ ]:

plt.plot(range(1, 8), error, color='red', linestyle='dashed', marker='o',

markerfacecolor='blue', markersize=10)

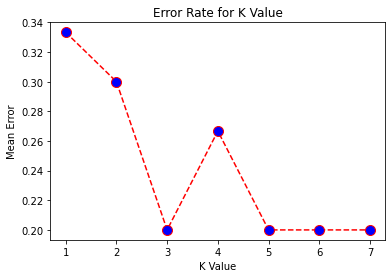
plt.title('Error Rate for K Value')

plt.xlabel('K Value')

plt.ylabel('Mean Error')

Out[ ]:

Text(0, 0.5, 'Mean Error')

****

In [ ]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

In [ ]:

names = ['f1', 'f2', 'f3', 'f4', 'Outcome']

df = pd.read\_csv('hw6\_data.csv')

df

Out[ ]:

|  | **f1** | **f2** | **f3** | **f4** | **Outcome** |
| --- | --- | --- | --- | --- | --- |
| **0** | 164 | 2.75 | 580 | 540 | N |
| **1** | 70 | 3.35 | 650 | 670 | Y |
| **2** | 273 | 3.77 | 530 | 520 | N |
| **3** | 16 | 3.37 | 530 | 460 | N |
| **4** | 77 | 2.92 | 550 | 530 | N |
| **...** | ... | ... | ... | ... | ... |
| **455** | 131 | 3.65 | 610 | 470 | Y |
| **456** | 302 | 2.73 | 550 | 430 | N |
| **457** | 172 | 4.03 | 460 | 460 | N |
| **458** | 29 | 3.55 | 600 | 720 | Y |
| **459** | 180 | 3.15 | 660 | 730 | N |

460 rows × 5 columns

In [ ]:

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

y = df.iloc[:, 4].values

In [ ]:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

In [ ]:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

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

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

In [ ]:

from sklearn.neighbors import KNeighborsClassifier

# n\_neighbors is k

classifier = KNeighborsClassifier(n\_neighbors=7)

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

Out[ ]:

KNeighborsClassifier(n\_neighbors=7)

In [ ]:

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

In [ ]:

from sklearn.metrics import classification\_report, confusion\_matrix

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

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

Accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", Accuracy)

[[18 14]

[16 44]]

precision recall f1-score support

N 0.53 0.56 0.55 32

Y 0.76 0.73 0.75 60

accuracy 0.67 92

macro avg 0.64 0.65 0.65 92

weighted avg 0.68 0.67 0.68 92

Accuracy: 0.6739130434782609

In [ ]:

error = []

# Calculating error for K values between 1 and 8

for i in range(1, 8):

knn = KNeighborsClassifier(n\_neighbors=i)

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

pred\_i = knn.predict(X\_test)

error.append(np.mean(pred\_i != y\_test))

In [ ]:

plt.plot(range(1, 8), error, color='red', linestyle='dashed', marker='o',markerfacecolor='blue', markersize=10)

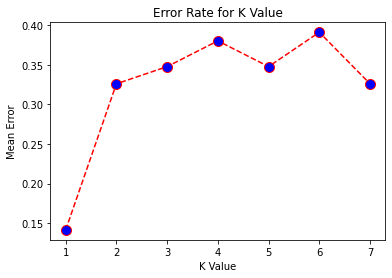
plt.title('Error Rate for K Value')

plt.xlabel('K Value')

plt.ylabel('Mean Error')

Out[ ]:

Text(0, 0.5, 'Mean Error')

****