K Nearest Neighbour

June 18, 2021

[14]:
```
%%capture
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from ipynb.fs.full.CleaningData import getDataset
from IPython.display import FileLink, FileLinks
from IPython.display import display, Markdown
from ipynb.fs.full.CleaningData import getCovarianceVector

import warnings
warnings.filterwarnings('ignore')

pd.set_option("display.max_rows", 100)
pd.set_option("display.max_columns", None)
```

[15]:
```
%%time
%%capture
df = getDataset(500)
```

```
Wall time: 8.72 s
```

[1]:
```
# df.head(10)
```

0.1 Getting Data

Using pandas and numpy to get train, validation and testing data

Also taking out the `URL` feature.

[17]:
```
#Deleting URL as the features have been extracted from the URL
del df['url']
```

[18]:
```
%%time
train, valid, test = np.split(df.sample(frac=1), [int(.6*len(df)), int(.
 ↪8*len(df))])

trainX = train.drop(['status'],1)
trainY = train['status']
```

```
validX = valid.drop(['status'],1)
validY = valid['status']

testX = test.drop(['status'],1)
testY = test['status']
```

```
Wall time: 29 ms
```

0.2 Fitting the Model

[19]:
```
from random import random
```

[20]:
```
#K-Nearest Neighbours
def KNN(trainXOrig, trainY, validX, k, isEuci):
    predictTp = []

    for tp in validX.values:
        #Copying the trainX so we don't use altered data sets
        trainX = trainXOrig.copy(deep=True)

        #Data Point - tp
        trainX = trainX.subtract(tp,axis=1)

        if (isEuci):
            #Euci
            trainX *= trainX

            #Get meanVec w/ Euci
            trainX['meanVec'] = np.sqrt(trainX.sum(axis=1).values)
        else:
            #Get meanVec w/ Manhatan
            trainX['meanVec'] = np.abs(trainX.sum(axis=1).values)

        #Sort meanVec
        trainX.sort_values('meanVec',inplace=True)

        #Counting the ones that are phising and the ones that are not
        countPhish = 0
        countLegi = 0

        #Getting unique values (0, 1) and the amount of times they showed up␣
↪for k
        countArr = np.unique(np.array([trainY[idx] for idx in trainX.index[:
↪k]]), return_counts=True)

        #Adding to Counter
        if (len(countArr[0]) == 1):
```

```
            if (countArr[0][0] == 1):
                countPhish += countArr[1][0]
            else:
                countLegi += countArr[1][0]
        elif (countArr[0][0] == 0):
            countLegi += countArr[1][0]
            countPhish += countArr[1][1]
        else:
            countPhish += countArr[1][0]
            countLegi += countArr[1][1]

        #Checking the highest
        if (countPhish > countLegi):
            predictTp.append(1)
        elif(countLegi > countPhish):
            predictTp.append(0)
        else:
            if (random() > 0.5):
                predictTp.append(1)
            else:
                predictTp.append(0)



    return predictTp
```

[21]:
```
def getConfusionMatrix(yActual, yObtained):
    if (len(yActual) != len(yObtained)):
        print("yActual and yObtained different lengths")
        return None

    phishMatch = 0
    phishNoMatch = 0
    legitMatch = 0
    legitNoMatch = 0

    for i in range(len(yActual)):
        yAct = yActual[i]
        yObt = yObtained[i]

        if (yAct == 1 and yObt == 1):
            phishMatch += 1
        if (yAct == 1 and yObt == 0):
            phishNoMatch += 1
        if (yAct == 0 and yObt == 0):
            legitMatch += 1
        if (yAct == 0 and yObt == 1):
```

```
            legitNoMatch += 1

    arr = np.array([[legitMatch,legitNoMatch],[phishNoMatch,phishMatch]])
    accu = (legitMatch + phishMatch)/sum([y for x in arr for y in x])

    df = pd.DataFrame(data=arr)

    df.columns = ["Legitimate", "Phishing"]
    df.index = ["Legitimate", "Phishing"]

    return df,accu
```

[22]:
```
def showResults(cf, accu, k):
    print("\n")

    print("============================")
    print("k is: " + str(k))
    print("============================\n")

    print("============================")
    print("Confusion Matrix\n")
    print(cf.head(2))
    print("\n========================\n")

    print("============================")
    print("Accuracy of: " + str(format(accu * 100,".5f")) + " %")
    print("============================")

    print("\n")
```

[23]:
```
def KNNwithOutput(trainXOrig, trainY, validX, validY, k, isEuci):
    yObtained = KNN(trainXOrig, trainY, validX, k, isEuci)

    confusionMat, accu = getConfusionMatrix(validY.values, yObtained)

    showResults(confusionMat, accu, k)

    return accu
```

0.3 Getting Optimal K - Eucilidean

[24]:
```
%%time

optiK = None
accu = 0.0
```

```
kLst = [3,5,7,9,15,27,51,67,99,217,515,999]

for k in kLst:
    newAccu = KNNwithOutput(trainX, trainY, validX, validY, k, True)

    if (newAccu > accu):
        accu = newAccu
        optiK = k
```

```
============================
k is: 3
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1087        62
Phishing            74      1073

========================

============================
Accuracy of: 94.07666 %
============================
```

```
============================
k is: 5
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1101        48
Phishing            85      1062

========================

============================
Accuracy of: 94.20732 %
============================
```

```
==========================
k is: 7
==========================

==========================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1107        42
Phishing            79      1068

=========================

==========================
Accuracy of: 94.72997 %
==========================




==========================
k is: 9
==========================

==========================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1104        45
Phishing            81      1066

=========================

==========================
Accuracy of: 94.51220 %
==========================




==========================
k is: 15
==========================

==========================
```

```
Confusion Matrix

            Legitimate  Phishing
Legitimate        1103        46
Phishing            95      1052

=========================

============================
Accuracy of: 93.85889 %
============================




============================
k is: 27
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1103        46
Phishing           116      1031

=========================

============================
Accuracy of: 92.94425 %
============================




============================
k is: 51
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1107        42
Phishing           137      1010

=========================
```

```
============================
Accuracy of: 92.20383 %
============================
```

```
============================
k is: 67
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1104        45
Phishing           153       994

========================

============================
Accuracy of: 91.37631 %
============================
```

```
============================
k is: 99
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1103        46
Phishing           160       987

========================

============================
Accuracy of: 91.02787 %
============================
```

```
============================
```

```
k is: 217
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1092        57
Phishing           204       943

========================

============================
Accuracy of: 88.63240 %
============================




============================
k is: 515
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1099        50
Phishing           284       863

========================

============================
Accuracy of: 85.45296 %
============================




============================
k is: 999
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1119        30
```

```
Phishing           441       706

==========================

============================
Accuracy of: 79.48606 %
============================


Wall time: 9min 10s
```

0.3.1 Optimal K - With Eucilidean

[25]:
```
print("============================")
print("Optimal k value is: " + str(optiK))
print("============================")
```

```
============================
Optimal k value is: 7
============================
```

[26]:
```
%%time
KNNwithOutput(trainX, trainY, testX, testY, optiK, True)
```

```
============================
k is: 7
============================

============================
Confusion Matrix

            Legitimate  Phishing
Legitimate        1104        40
Phishing            79      1074

==========================

============================
Accuracy of: 94.81933 %
============================


Wall time: 45.7 s
```

[26]:
```
0.9481932956029604
```