Michael Reilly

Professor Wang

CS-559-A

11/30/20

I pledge my honor that I have abided by the Stevens Honor System.

Problem 1. (60 points) Download the "Pima Indians Diabetes Database" from Canvas.

(a) Implement a classifier using Maximum-Likelihood Estimation that takes into account features 2 to 4, among the 8 available features.

(b) Train the classifier on the same samples and run them 10 or more times. Record the mean and standard deviation of the accuracy. Use 50% of the data for training and the rest for testing. Make sure that the two sets are disjoint.

(c) Submit code, but not data, taking into account the assumptions made.

*Can be found in zip file as Reilly_HW2_Q1.py,

*Run in command line by typing: python3 Reilly_HW2_Q1.py

```
# Michael Reilly
# I pledge my honor that I have abided by the Stevens Honor System.

import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
import math

# Command line usage: python3 Reilly_HW2_Q1.py

# A classifier using the Maximum Likelihood Estimator is the bayes
classifier
# In this case we'll use the importable Gaussian Naive Bayes classifier
# Got the idea to use this from
https://towardsdatascience.com/bayes-classifier-with-maximum-likelihood-es
timation-4b754b641488
```

```python
# Imported the scikit_learn Gaussian Naive Bayes Classifier in python

def mean(list):
    added=0
    i=0
    while i<10:
        added+=list[i]
        i+=1
    return added/10

def stdev(list):
    standard_deviation=0
    i=0
    while i<10:
        standard_deviation+=(list[i]-mean(list))**2
        i+=1
    std_dev=standard_deviation/9
    std_deviation=math.sqrt(std_dev)
    return std_deviation

csv=pd.read_csv("pima-indians-diabetes.csv")
columns=csv.iloc[:,1:4]
classifier=csv.iloc[:,-1]
# Documentation for GaussianNB():
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
MLE=GaussianNB()
accuracy=[]
i=1

while i<=10:
    x_train,x_test,y_train,y_test=train_test_split(
    columns,classifier,test_size=0.5)
    MLE.fit(x_train,y_train)
    accuracy.append(MLE.score(x_test,y_test))
    i+=1

print("Mean: ",mean(accuracy))
print("Standard Deviation: ",stdev(accuracy))
```

Problem 2. (40 points) Use the “Pima Indians Diabetes Database” and implement a k-Nearest Neighbor classifier. Split the data in half to form the training and test sets and use features 2 to 4 as above. Report mean accuracy for k=1, 5 and 11, as well as its standard deviation, over at least 10 trials for each value of k.

*Can be found in zip file as Reilly_HW2_Q2.py,

*Run in command line by typing: python3 Reilly_HW2_Q2.py

```
# Michael Reilly
# I pledge my honor that I have abided by the Stevens Honor System.

import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import math

# Command line usage: python3 Reilly_HW2_Q2.py

# Imported the scikit_learn KNeighborsClassifier for KNN classifiers in
python

def mean(list):
    added=0
    i=0
    while i<10:
        added+=list[i]
        i+=1
    return added/10

def stdev(list):
    standard_deviation=0
    i=0
    while i<10:
        standard_deviation+=(list[i]-mean(list))**2
        i+=1
    std_dev=standard_deviation/9
    std_deviation=math.sqrt(std_dev)
    return std_deviation
```

```python
csv=pd.read_csv("pima-indians-diabetes.csv")
columns=csv.iloc[:,1:4]
classifier=csv.iloc[:,-1]
accuracy1=[]
accuracy5=[]
accuracy11=[]
i=1

while i<=10:
    x_train,x_test,y_train,y_test=train_test_split(
    columns,classifier,test_size=0.5)
    # Documentation for KNeighborsClassifier():
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeigh
borsClassifier.html
    knn1=KNeighborsClassifier(n_neighbors=1)
    knn5=KNeighborsClassifier(n_neighbors=5)
    knn11=KNeighborsClassifier(n_neighbors=11)
    knn1.fit(x_train,y_train)
    knn5.fit(x_train,y_train)
    knn11.fit(x_train,y_train)
    accuracy1.append(knn1.score(x_test,y_test))
    accuracy5.append(knn5.score(x_test,y_test))
    accuracy11.append(knn11.score(x_test,y_test))
    i+=1

print("For k=1")
print("Mean: ",mean(accuracy1))
print("Standard Deviation: ",stdev(accuracy1))
print("\n")
print("For k=5")
print("Mean: ",mean(accuracy5))
print("Standard Deviation: ",stdev(accuracy5))
print("\n")
print("For k=11")
print("Mean: ",mean(accuracy11))
print("Standard Deviation: ",stdev(accuracy11))
```