Tae Coding
Introduction to Data Science: CS61
Summer 2018
Homework#7

Date Given: July 3, 2018 Due Date:

==

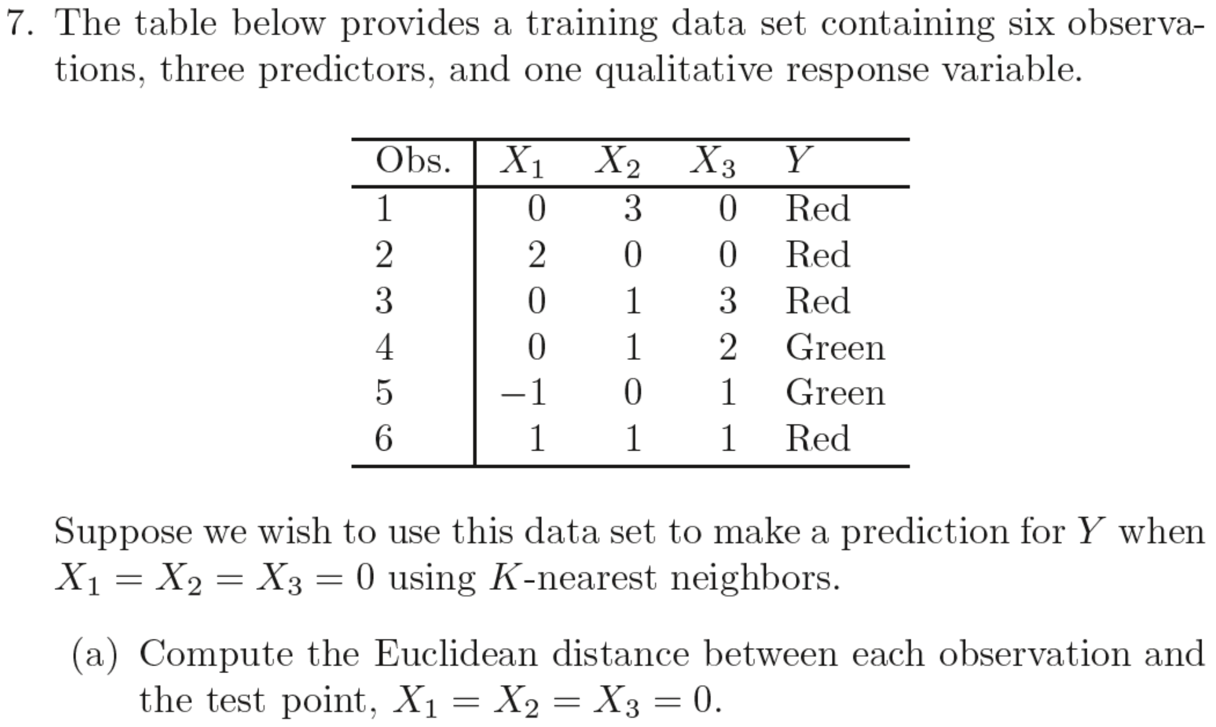
There are 2 problems in this homework assignment. Please use Python’s Scikit-Learn package to solve these 2 problems.

Text Book: “An Introduction to Statistical Learning” (ISLR).
By James, Witten, Hastie, Tibshirani
Chapter 2: Statistical Learning: Page 53/54, Problem#7.

There is no need to buy this text book. I have copied the problems from the PDF version of this book.

===

Problem#1

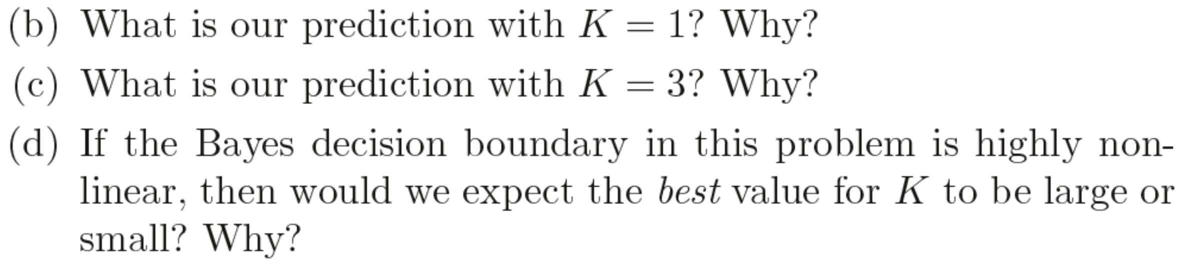


7. The table below provides a training data set containing six observations, three predictors, and one qualitative response variable.

Obs.	X_1	X_2	X_3	Y
1	0	3	0	Red
2	2	0	0	Red
3	0	1	3	Red
4	0	1	2	Green
5	-1	0	1	Green
6	1	1	1	Red

Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K-nearest neighbors.

(a) Compute the Euclidean distance between each observation and the test point, $X_1 = X_2 = X_3 = 0$.



(b) What is our prediction with $K = 1$? Why?

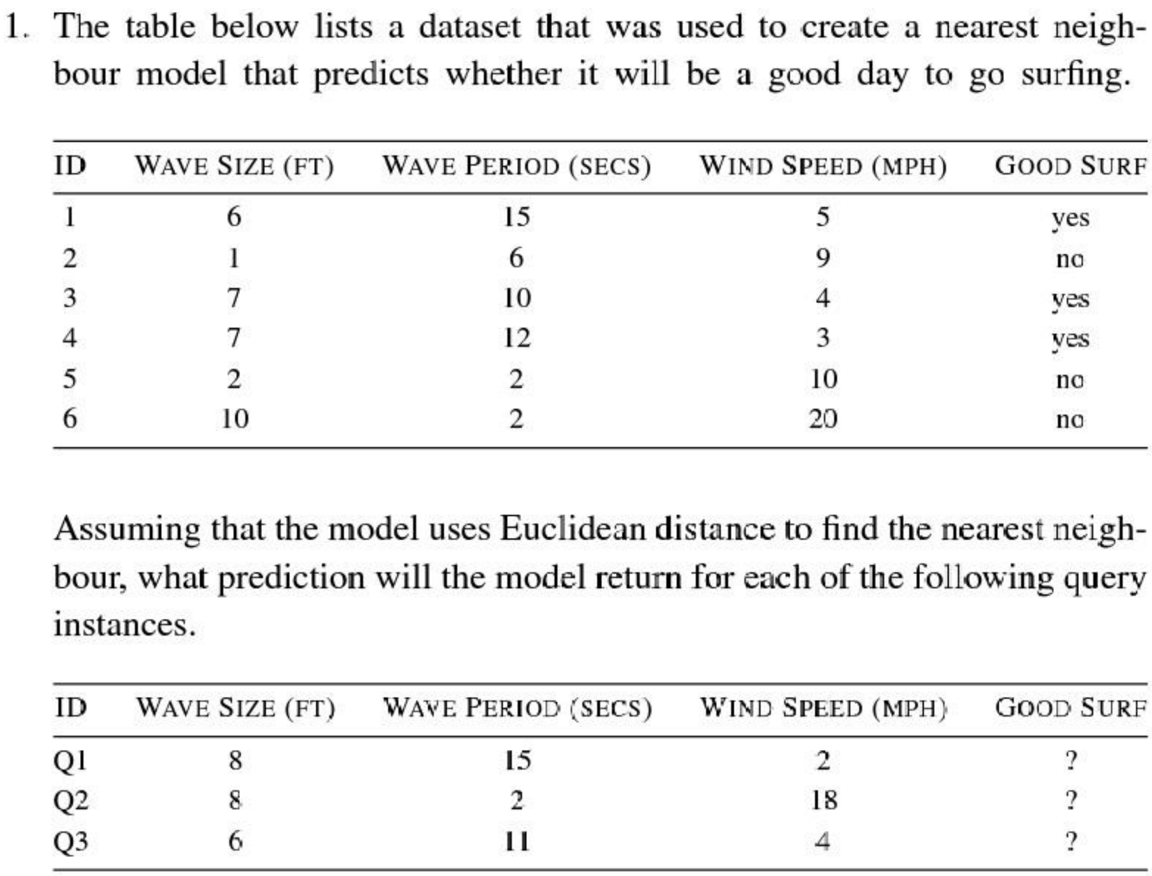
(c) What is our prediction with $K = 3$? Why?

(d) If the Bayes decision boundary in this problem is highly nonlinear, then would we expect the *best* value for K to be large or small? Why?

Book: Fundamentals of Machine Learning for Predictive Data Analytics
By: Kelleher, MacNamee, D’Arcy

Chapter 5: Similarity-based Learning: Page 240: Problem#1
==

Problem#2



1. The table below lists a dataset that was used to create a nearest neighbour model that predicts whether it will be a good day to go surfing.

ID	WAVE SIZE (FT)	WAVE PERIOD (SECS)	WIND SPEED (MPH)	GOOD SURF
1	6	15	5	yes
2	1	6	9	no
3	7	10	4	yes
4	7	12	3	yes
5	2	2	10	no
6	10	2	20	no

Assuming that the model uses Euclidean distance to find the nearest neighbour, what prediction will the model return for each of the following query instances.

ID	WAVE SIZE (FT)	WAVE PERIOD (SECS)	WIND SPEED (MPH)	GOOD SURF
Q1	8	15	2	?
Q2	8	2	18	?
Q3	6	11	4	?

Problem#1: Simple Python Code

```
import numpy as np
import pandas as pd
from collections import Counter
###############################################
# Read the Training and Test dataset

train = pd.read_csv("P1 Book C2 - P7 - Train.csv")
test = pd.read_csv("P1 Book C2 - P7 - Test.csv")

print(train)
   Unnamed: 0  X1  X2  X3      Y
0           1   0   3   0    Red
1           2   2   0   0    Red
2           3   0   1   3    Red
3           4   0   1   2  Green
4           5  -1   0   1  Green
5           6   1   1   1    Red

print(test)
   Unnamed: 0  X1  X2  X3   Y
0           1   0   0   0 NaN

################################################
# Compute the distance
# from Test object to all the Train's objects
trainC = train.shape[0]
print(trainC)
6

sum = np.zeros(trainC)
for i in range (0, trainC):
    #print(i)
    sum[i] = sum[i] + (train.X1[i] - test.X1[0])**2
    sum[i] = sum[i] + (train.X2[i] - test.X2[0])**2
    sum[i] = sum[i] + (train.X3[i] - test.X3[0])**2

distance = np.sqrt(sum)
print(sum)
[  9.   4.  10.   5.   2.   3.]
print(distance)
[ 3.          2.          3.16227766  2.23606798  1.41421356  1.73205081]

train['dist'] = distance
print(train)
   Unnamed: 0  X1  X2  X3      Y      dist
0           1   0   3   0    Red  3.000000
1           2   2   0   0    Red  2.000000
2           3   0   1   3    Red  3.162278
3           4   0   1   2  Green  2.236068
4           5  -1   0   1  Green  1.414214
5           6   1   1   1    Red  1.732051
```

```
###########################################
# Sort the dataset by distance

trainSorted = train.sort_values(['dist'])

print(trainSorted)
   Unnamed: 0  X1  X2  X3      Y      dist
4           5  -1   0   1  Green  1.414214
5           6   1   1   1    Red  1.732051
1           2   2   0   0    Red  2.000000
3           4   0   1   2  Green  2.236068
0           1   0   3   0    Red  3.000000
2           3   0   1   3    Red  3.162278

#############################################
# Find the nearest neighbor

k = 1
nearestNeighbor = trainSorted.Y[0:k]
print(nearestNeighbor)
4    Green
Name: Y, dtype: object

Counter(nearestNeighbor)
Out[36]: Counter({'Green': 1})



k = 3
nearestNeighbor = trainSorted.Y[0:k]
print(nearestNeighbor)
4    Green
5      Red
1      Red
Name: Y, dtype: object

Counter(nearestNeighbor)
Out[40]: Counter({'Green': 1, 'Red': 2})
```

Part (d) For non-linear Bayes boundary, small value of ‘k’ would be better. Small value of ‘k’ would be able to capture the irregular boundary

Problem#1: Scikit-Learn

```
###############################################
# Read the Training and Test dataset
#

train = pd.read_csv("P1 Book C2 - P7 - Train.csv")
test = pd.read_csv("P1 Book C2 - P7 - Test.csv")

print(train)
   Unnamed: 0  X1  X2  X3      Y
0           1   0   3   0    Red
1           2   2   0   0    Red
2           3   0   1   3    Red
3           4   0   1   2  Green
4           5  -1   0   1  Green
5           6   1   1   1    Red

print(test)
   Unnamed: 0  X1  X2  X3   Y
0           1   0   0   0 NaN

X_train = np.array(train[['X1','X2','X3']])
X_train
Out[12]:
array([[ 0,  3,  0],
       [ 2,  0,  0],
       [ 0,  1,  3],
       [ 0,  1,  2],
       [-1,  0,  1],
       [ 1,  1,  1]], dtype=int64)

y_train = train['Y']
y_train
Out[14]:
0      Red
1      Red
2      Red
3    Green
4    Green
5      Red
Name: Y, dtype: object

X_test = np.array(test[['X1','X2','X3']])

X_test
Out[16]: array([[0, 0, 0]], dtype=int64)
```

```
#####################################################
clf = neighbors.KNeighborsClassifier(n_neighbors=1)
clf.fit(X_train, y_train)
Out[19]: 
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')

clf.predict(X_test)
Out[20]: array(['Green'], dtype=object)



#####################################################
clf = neighbors.KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
Out[23]: 
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')

clf.predict(X_test)
Out[24]: array(['Red'], dtype=object)
```

Problem#2: Simple Python Code

```
import numpy as np
import pandas as pd
from collections import Counter
########################################################
# Read dataset - Train + Test
train = pd.read_csv("P2 Book P1 Surfing Train.csv")
test = pd.read_csv("P2 Book P1 Surfing Test.csv")

train.columns = ['ID','WaveSize', 'WavePeriod', 'WindSpeed','GoodSurf']
test.columns = ['ID','WaveSize', 'WavePeriod', 'WindSpeed']

print(train)
   ID  WaveSize  WavePeriod  WindSpeed GoodSurf
0   1         6          15          5      yes
1   2         1           6          9       no
2   3         7          10          4      yes
3   4         7          12          3      yes
4   5         2           2         10       no
5   6        10           2         20       no

print(test)
   ID  WaveSize  WavePeriod  WindSpeed
0   1         8          15          2
1   2         8           2         18
2   3         6          11          4
##########################################################
# Compute the distance
trainC = train.shape[0]
sum = np.zeros(trainC)
for i in range (0, trainC):
    #print(i)
    sum[i] = sum[i] + (train.WaveSize[i] - test.WaveSize[0])**2
    sum[i] = sum[i] + (train.WavePeriod[i] - test.WavePeriod[0])**2
    sum[i] = sum[i] + (train.WindSpeed[i] - test.WindSpeed[0])**2

distance = np.sqrt(sum)
print(sum)
[  13.  179.   30.   11.  269.  497.]
print(distance)
[  3.60555128  13.37908816   5.47722558   3.31662479  16.40121947
  22.29349681]
train['dist'] = distance
print(train)
   ID  WaveSize  WavePeriod  WindSpeed GoodSurf       dist
0   1         6          15          5      yes   3.605551
1   2         1           6          9       no  13.379088
2   3         7          10          4      yes   5.477226
3   4         7          12          3      yes   3.316625
4   5         2           2         10       no  16.401219
5   6        10           2         20       no  22.293497
```

```
#################################################
# Sort the distance
trainSorted = train.sort_values(['dist'])

print(trainSorted)
   ID  WaveSize  WavePeriod  WindSpeed GoodSurf       dist
3   4         7          12          3      yes   3.316625
0   1         6          15          5      yes   3.605551
2   3         7          10          4      yes   5.477226
1   2         1           6          9       no  13.379088
4   5         2           2         10       no  16.401219
5   6        10           2         20       no  22.293497



#################################################
#  Find the nearest neighbor
#

k = 3

nearestNeighbor = trainSorted.GoodSurf[0:k]

print(nearestNeighbor)
3    yes
0    yes
2    yes
Name: GoodSurf, dtype: object


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

This code predicts ‘Good Surf’ for only the first test data.

```
print(test)
   ID  WaveSize  WavePeriod  WindSpeed GoodSurf
0   1         8          15          2      yes
```

Answer is : Yes

Problem#2: Scikit-Learn

```
import numpy as np
import pandas as pd
from sklearn import neighbors
###############################################
# Read the Training and Test dataset

train = pd.read_csv("P2 Book P1 Surfing Train.csv")
test = pd.read_csv("P2 Book P1 Surfing Test.csv")
train.columns = ['ID','WaveSize', 'WavePeriod', 'WindSpeed','GoodSurf']
test.columns = ['ID','WaveSize', 'WavePeriod', 'WindSpeed']

print(train)
   ID  WaveSize  WavePeriod  WindSpeed GoodSurf
0   1         6          15          5      yes
1   2         1           6          9       no
2   3         7          10          4      yes
3   4         7          12          3      yes
4   5         2           2         10       no
5   6        10           2         20       no

print(test)
   ID  WaveSize  WavePeriod  WindSpeed
0   1         8          15          2
1   2         8           2         18
2   3         6          11          4

X_train = np.array(train[['WaveSize', 'WavePeriod', 'WindSpeed']])
X_train
Out[14]:
array([[ 6, 15,  5],
       [ 1,  6,  9],
       [ 7, 10,  4],
       [ 7, 12,  3],
       [ 2,  2, 10],
       [10,  2, 20]], dtype=int64)

y_train = train['GoodSurf']
y_train
Out[16]:
0    yes
1     no
2    yes
3    yes
4     no
5     no
Name: GoodSurf, dtype: object
```

```
X_test = np.array(test[['WaveSize', 'WavePeriod', 'WindSpeed']])
X_test
Out[18]:
array([[ 8, 15,  2],
       [ 8,  2, 18],
       [ 6, 11,  4]], dtype=int64)

#####################################################
clf = neighbors.KNeighborsClassifier(n_neighbors=3)

clf.fit(X_train, y_train)
Out[21]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')

clf.predict(X_test)
Out[22]: array(['yes', 'no', 'yes'], dtype=object)

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


   ID  WaveSize  WavePeriod  WindSpeed GoodSurf
0   1         8          15          2      yes
1   2         8           2         18       no
2   3         6          11          4      yes
```