National University of Computer & Emerging Sciences, Karachi

Spring 2019 CS-Department

Midterm # 2, 1st April 2018, 9:00 am – 11:00 am

Course Code: CS481	**Course Name:** Data Science
Instructor Name: Dr Muhammad Atif Tahir	
Student Roll No:	**Section No:**

Instructions:

- Return the question paper.
- You are allowed to use PCs but all programs should be written in the answer sheet
- Read each question completely before answering it. There are 3 **questions and 2 pages**

Time: 120 minutes. **Max Marks**: 12.5 points

Question 1 [2.5 Points]: Complete the following program

import pandas as pd

# Create a dataframe object "df" with the following values. Either through creating file or parameters [0.5 Points]

```
    Data Science Machine Learning Deep Learning  Average
0  UnderGraduate              NaN           NaN     70.1
1            NaN          Masters           NaN     70.5
2            NaN              NaN       Masters     70.3
```

# Create following dataframe from df [Hint: you may like to use melt() function in Pandas [2 Points]

```
   Average            Course          Level
0     70.1      Data Science  UnderGraduate
1     70.5  Machine Learning        Masters
2     70.3     Deep Learning        Masters
```

Question2 [5 Points]: Read the attached algorithm from paper with title "Weighted k-Nearest-Neighbor Techniques and Ordinal Classification". This paper discusses weighted kNN classifier. You need to implement pseudocode given on Appendix using Python without using any libraries of kNN classifier from sklearn. Test your program using following data points from Table 2. Use k = 4, manhattan distance: $d(p,q) = \sum_i |p_i - q_i|$.
, and compute kernel using inversion kernel (1/|d|). An example is given below that can help you in the understanding of the algorithm. [Hint: you may like to use DistanceMetric class from sklearn.neighbors. Use 'manhattan' distance metric]

Instance #	Att1	Att2	Actual Class
1	2	3	0
2	1	5	1
3	4	2	1
4	2	5	0
5	6	8	0

Instance #	Att1	Att2	Predicted Class
6	3	1	?
7	2	2	?

Table2: Training and Test Data for Question 2.

For test sample 6

D1 = |3-2| + |1-3| = 1 + 2 = 3; W1 = 3 / 10 = 0.3, K(x,x(1)) = 1 / 0.3 = 3.34

D2 = |3-1| + |1-5| = 2 + 4 = 6, W2 = 6/10 = 0.6, K(x,x(2)) = 1/0.6 = 1.67

D3 = |3-4| + |1-2| = 1 + 1 = 2, W3 = 0.2, K(x,x(3)) = 1/0.2 = 5

D4 = |3-2| + |1-5| = 1 + 4 = 5, W4 = 0.5, K(x,x(4)) = 1/0.5 = 2

D5 = |3-6| + |1-8| = 3 + 7 = 10, W5 = 1,

For 0; 3.34 + 2 = 5.34 and For 1; 1.67+5 = 6.67 thus belongs to 1

For test sample 7

D1 = |2-2| + |2-3| = 0 + 1 = 1; W1 = 1 / 10 = 0.1, K(x,x(1)) = 1 / 0.1 = 10
D2 = |2-1| + |2-5| = 1 + 3 = 4, W2 = 4/10 = 0.6, K(x,x(2)) = 1/0.4 = 2.5
D3 = |2-4| + |2-2| = 2 + 0 = 2, W3 = 0.2, K(x,x(3)) = 1/0.2 = 5
D4 = |2-2| + |2-5| = 0 + 3 = 3, W4 = 0.3, K(x,x(4)) = 1/0.3 = 3.33
D5 = |2-6| + |2-8| = 3 + 7 = 10, W5 = 1,

For 0; 10 + 3.33 = 13.33 and For 1; 2.5+5 = 7.5 thus belongs to 0

Question3 [5 Points]: Complete the following program. This program will evaluate SVM with all original features, SVM with features obtained from Principal Component Analysis (PCA) only and SVM with original features plus PCA. Accuracies need to be reported using 5-Fold CV. Use help to read more information about PCA. For PCA, transform data into 2 dimensions.

```
import numpy as np
import pandas as pd
from sklearn import datasets
#from sklearn.model_selection import KFold
from sklearn.cross_validation import KFold
from sklearn import cross_validation
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA

# load digits dataset
digits = datasets.load_digits()

# print the number of samples and number of attributes
# expected output is shown below
```

```
The number of instances are:  1797
The number of attributes are:  64
Fold1: Accuracy using SVM: 0.9638888888888889
Fold2: Accuracy using SVM: 0.9222222222222223
Fold3: Accuracy using SVM: 0.9637883008356546
Fold4: Accuracy using SVM: 0.9637883008356546
Fold5: Accuracy using SVM: 0.9303621169916435


Fold1: Accuracy using SVM and with only PCA features: 0.6138888888888889
Fold2: Accuracy using SVM and with only PCA features: 0.6
Fold3: Accuracy using SVM and with only PCA features: 0.6128133704735376
Fold4: Accuracy using SVM and with only PCA features: 0.6155988857938719
Fold5: Accuracy using SVM and with only PCA features: 0.5738161559888579


Fold1: Accuracy using SVM and added PCA features in whole data: 0.9611111111111111
Fold2: Accuracy using SVM and added PCA features in whole data: 0.925
Fold3: Accuracy using SVM and added PCA features in whole data: 0.9637883008356546
Fold4: Accuracy using SVM and added PCA features in whole data: 0.9610027855153204
Fold5: Accuracy using SVM and added PCA features in whole data: 0.924791086350975
```

BEST OF LUCK!