Data Driven Computing and Networking (DDCN-2019)

Classification using K-NN Algorithm

A. Write a python script to perform the following tasks:-

1. **Load all the required packages to implement KNN Classification algorithm**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

2. **Load Iris flower data set using read_csv method and store it in a variable "df"**

```
names=['sepal_length','sepal_width','petal_length','petal_width','class']
df=pd.read_csv('C:/Users/pc/Desktop/DDCN
2019/Classification/Datasets/IRIS.csv',header=None,names=names)
print (df.head())
```

3. **Create design matrix X and target vector Y from the variable "df". The "class" column represents the target vector.**

```
X=np.array(df.ix[:,0:4])
Y=np.array(df['class'])
```

4. **Split the data stored in variable "df" into training sets (X_train, Y_train) and test sets (X_test, Y_test).**

```
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.33)
```

5. **Instantiate KNeighborsClassifier method with three number of neighbors and store its output in variable knn, representing a classifier model.**

```
knn=KNeighborsClassifier(n_neighbors=3)
```

6. **Fit kmeans classifer model on training data set of design matrix and target vector X.**

```
knn.fit(X_train,Y_train)
```

7. **Predict the target vector using test data set of the design matrix X_test.**

```
pred=knn.predict(X_test)
```

8. **Evaluate accuracy of the predicted value by comparing the predicted target value and actual target value from the test data set Y_test.**

```
print( ' Accuracy score is',accuracy_score(Y_test, pred))
```

9. **Create odd list of knn neighbors in range 1 to 50 and store it in a variable myList.**

```
myList=list(range(1,50))
# subsetting just the odd ones
neighbors=filter(lambda x:x%2!=0, myList)
```

10. **Perform 10-fold cross validation for each value of the neighbors and store cross validation score in a list cv_score.**

```
# Defining empty list for holding cvscores
cv_scores=[]
# perform 10-fold cross validation
for k in neighbors:
    knn=KNeighborsClassifier(n_neighbors=k)
    scores=cross_val_score(knn,X_train,Y_train,cv=10,scoring='accuracy')
    cv_scores.append(scores.mean())
```

11. **Compute MSE for each value of the list cv_score.**

```
MSE=[1-x for x in cv_scores]
```

12. **Determine best value of k and print optimal number of neighbors.**

```
# Determining best k
optimal_k=neighbors[MSE.index(min(MSE))]
print (' The optimal number of neighbors is %d', optimal_k)
```

13. **Plot Misclassification error and all values of k.**

```
plt.plot(neighbors,MSE)
```

14. **Assign the x axis label as " Number of Neighbors k" and y axis label as " Misclassification Error" and display the scatter chart.**

```
plt.xlabel('Number of Neighbors k')
plt.ylabel('Misclassification Error')
plt.show()


print ("locals")
locals()
print ("globals")
globals()
```