Assignment-02 Date: 08/08/2020

Title : Clustering.

Problem : Consider a suitable dataset for clustering
Statement of data instances in different groups, apply different clustering techniques (min 2) Visualize clusters using suitable tools.

S/W & H/W : Rstudio / Jupyter Notebook.
Requirements PIV, 2GB RAM, 500 GB HDD.

Learning Objectives: Use R functions / Scikit-learn functions to create K-means clustering models & heirachical clustering models.

Learning Outcomes: Visualize the effects of K means & heirarchical clustering using graphic capabilities.

Theory :-

I] K-means Clustering

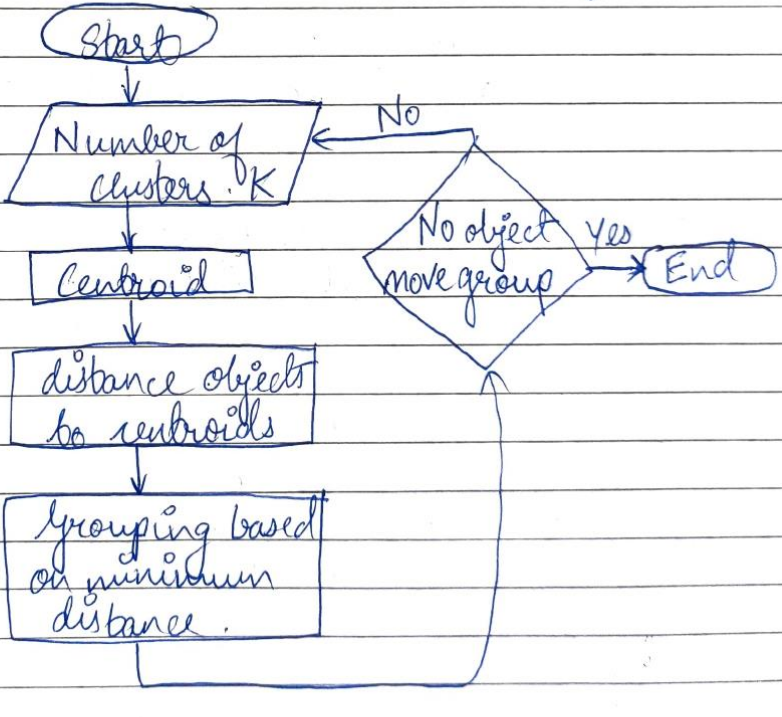
- It is a type of unsupervised learning, which is used when you have unlabelled data.
- The goal of this algorithm is to find groups in the data, with number of groups represented by the variable K.
- The algorithm works iteratively to assign each data point to one of K group based on features that are provided.
- Data points are clustered based on feature similarity.
- The results of K means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data.

2. Labels for training data (each data point is assigned to a single cluster).

- Rather than defining groups before looking at the data, clustering allows you to find & analyze the groups that have been formed organically.

→ Steps to perform K-means Clustering.

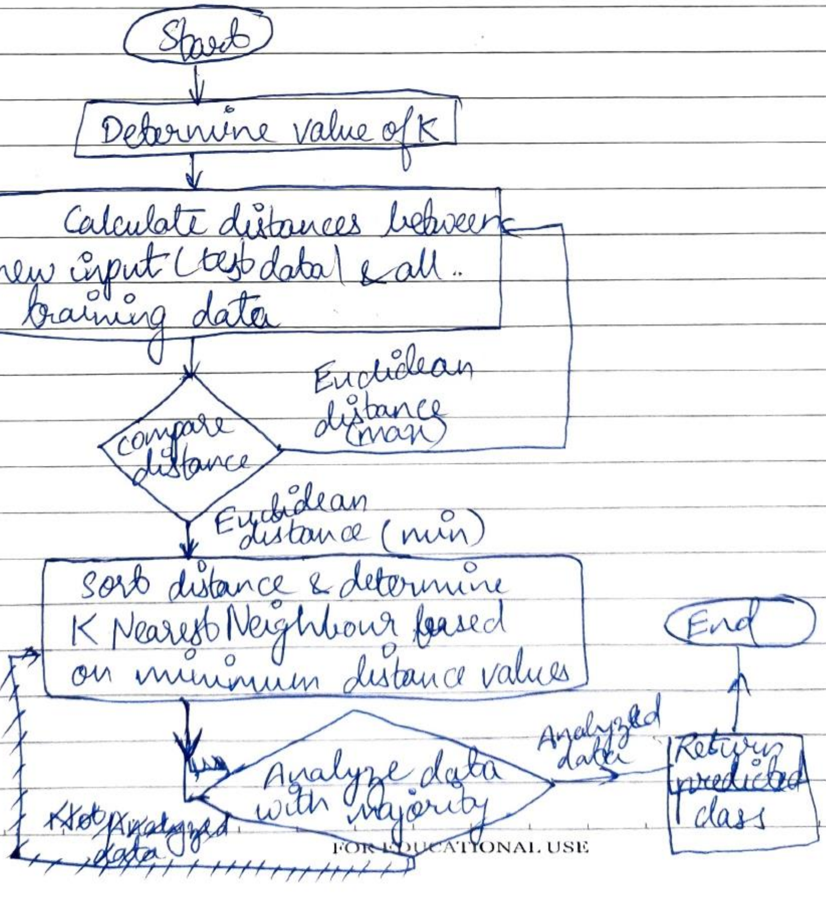

Start
Number of clusters K
Centroid
distance objects to centroids
Grouping based on minimum distance.
No object move group
No
Yes
End

B] K Nearest Neigbour (KNN) clustering.

- It is a supervised classification algorithm.
- It takes bunch of labeled points and uses them to learn how to label other points. ~~To label~~ a

VS

- To label a new point, it looks at the labeled points closest to that new point which are its nearest neighbors, and has those neighbors vote.
- So whichever label, the most of the neighbours have is the label, the most of the neighbors have is the label for the new point.
- Here 'K' in KNN is number of neighbors it checks.
- It is supervised because you are trying to classify a point based on the known classification of other points.


Start
Determine value of K
Calculate distances between new input (test data) & all .. training data
Euclidean distance (max)
compare distance
Euclidean distance (min)
sort distance & determine K Nearest Neighbour based on minimum distance values
Analyze data with majority
Analyzed data
Return predicted class
End
Not Analyzed data

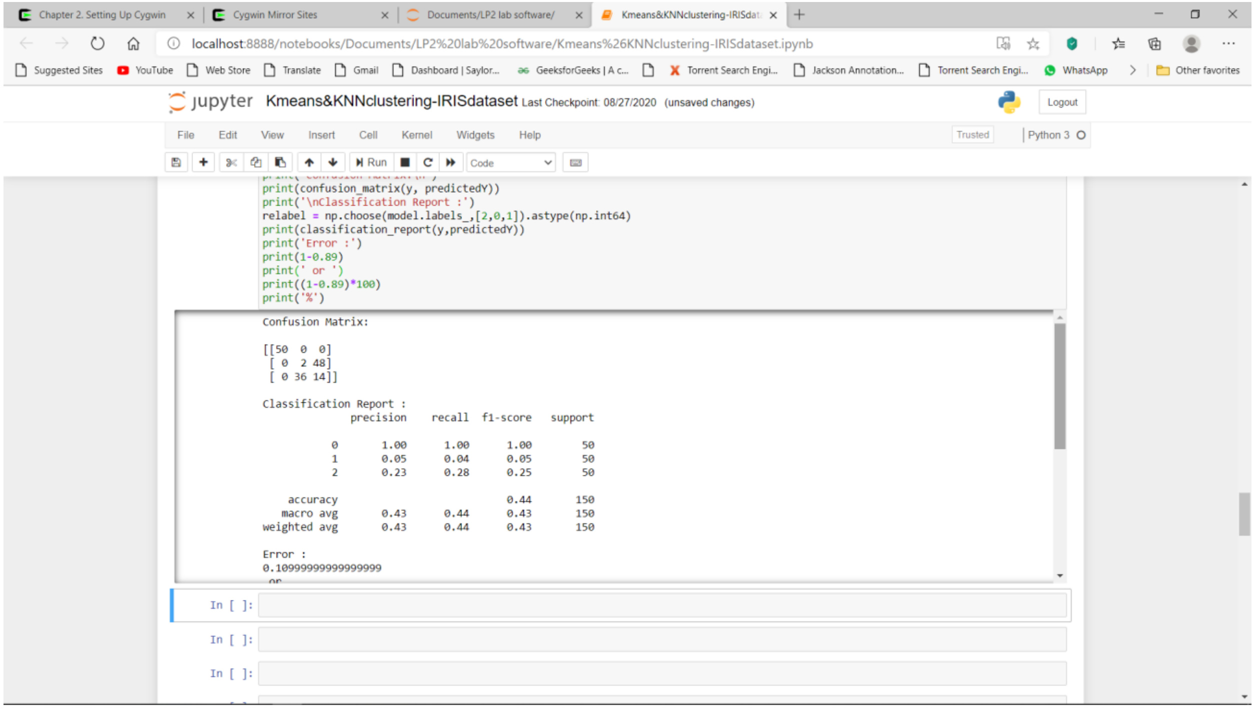
Test cases

Sr no	Description	Expected O/P	Actual O/P
1)	In KNN clustering method (a un-supervised algorithm) we created confusion matrix & Classification report based on Euclidean distance K clusters are formed.	No. of clusters rendered = 5	success
2)	Visuals cluster using single, complete & average linkages.	clusters displayed by means of scatter plot	success
3)	While fitting K means to dataset, put random state = 42	success	success

Conclusion: Hence, we have successfully implemented hierarchical clustering and K means clustering algorithm in python using jupyter notebooks.

Output:

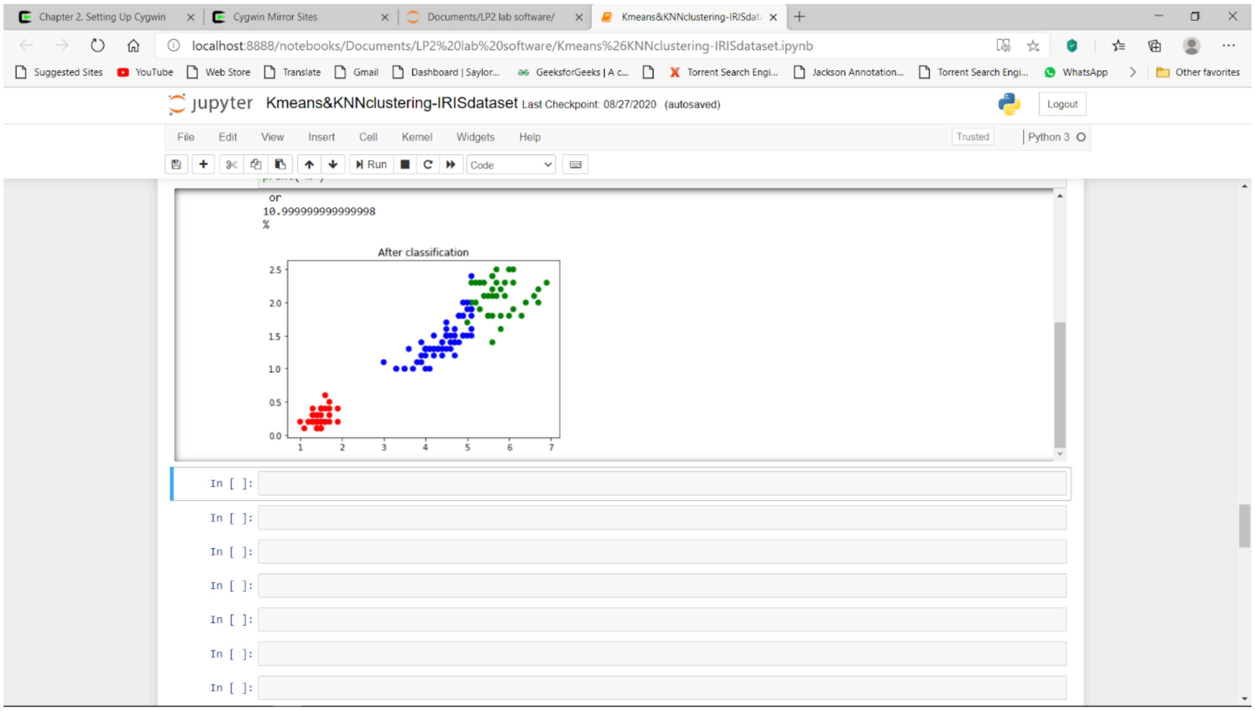
1)K-Means


localhost:8888/notebooks/Documents/LP2%20lab%20software/Kmeans%26KNNclustering-IRISdataset.ipynb
jupyter Kmeans&KNNclustering-IRISdataset Last Checkpoint: 08/27/2020 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
print(confusion_matrix(y, predictedY))
print('\nClassification Report :')
relabel = np.choose(model.labels_,[2,0,1]).astype(np.int64)
print(classification_report(y,predictedY))
print('Error :')
print(1-0.89)
print(' or ')
print((1-0.89)*100)
print('%')
```

```
Confusion Matrix:

[[50  0  0]
 [ 0  2 48]
 [ 0 36 14]]

Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.05      0.04      0.05        50
           2       0.23      0.28      0.25        50

    accuracy                           0.44       150
   macro avg       0.43      0.44      0.43       150
weighted avg       0.43      0.44      0.43       150

Error :
0.10999999999999999
```

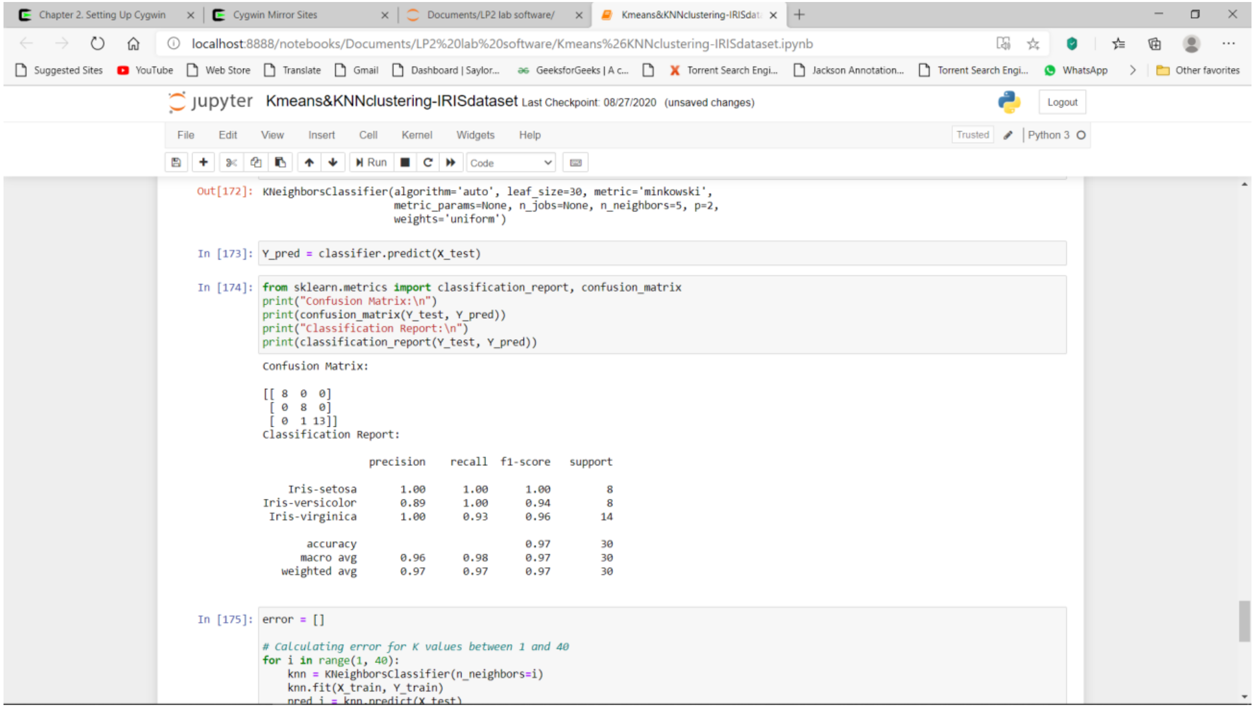
```
In [ ]:
In [ ]:
In [ ]:
```


localhost:8888/notebooks/Documents/LP2%20lab%20software/Kmeans%26KNNclustering-IRISdataset.ipynb
jupyter Kmeans&KNNclustering-IRISdataset Last Checkpoint: 08/27/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
After classification
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

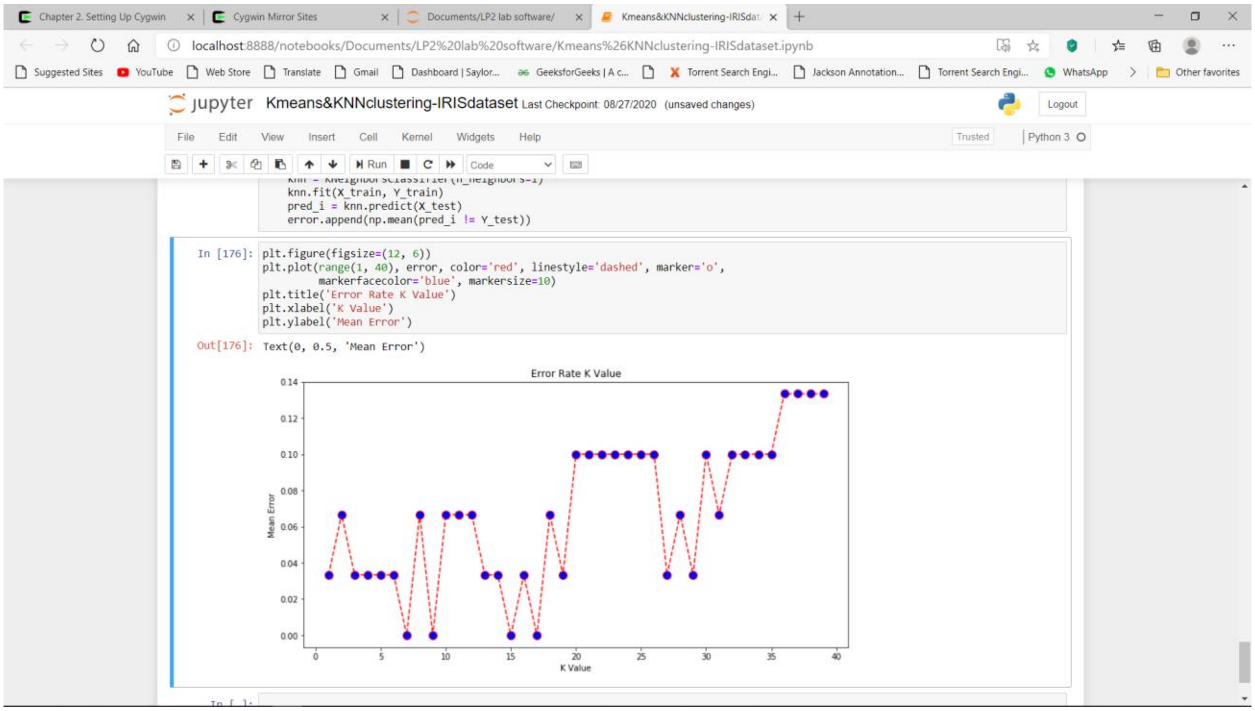
```
 or
10.999999999999998
%
```

```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```

2)KNN



```
Out[172]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                               weights='uniform')

In [173]: Y_pred = classifier.predict(X_test)

In [174]: from sklearn.metrics import classification_report, confusion_matrix
          print("Confusion Matrix:\n")
          print(confusion_matrix(Y_test, Y_pred))
          print("Classification Report:\n")
          print(classification_report(Y_test, Y_pred))

          Confusion Matrix:

          [[ 8  0  0]
           [ 0  8  0]
           [ 0  1 13]]
          Classification Report:

                           precision    recall  f1-score   support

              Iris-setosa       1.00      1.00      1.00         8
          Iris-versicolor       0.89      1.00      0.94         8
           Iris-virginica       1.00      0.93      0.96        14

                 accuracy                           0.97        30
                macro avg       0.96      0.98      0.97        30
             weighted avg       0.97      0.97      0.97        30


In [175]: error = []

          # Calculating error for K values between 1 and 40
          for i in range(1, 40):
              knn = KNeighborsClassifier(n_neighbors=i)
              knn.fit(X_train, Y_train)
              pred_i = knn.predict(X_test)
```



```
              knn = KNeighborsClassifier(n_neighbors=i)
              knn.fit(X_train, Y_train)
              pred_i = knn.predict(X_test)
              error.append(np.mean(pred_i != Y_test))

In [176]: plt.figure(figsize=(12, 6))
          plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
                   markerfacecolor='blue', markersize=10)
          plt.title('Error Rate K Value')
          plt.xlabel('K Value')
          plt.ylabel('Mean Error')

Out[176]: Text(0, 0.5, 'Mean Error')
```