

K-Means Clustering

Implementation of the k-means clustering algorithm to cluster a data set. Used a data set from the UC Irvine Machine Learning Repository at: (<http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>)

Format of data file

The data file that is being clustered is a database related to iris plants. A complete description can be found here: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names>. Will use the file at <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data> as our input file. Each line of the csv file looks something like this: 5.1,3.5,1.4,0.2,Iris-setosa It consists of four floating point values and a text label for the type of iris plant. The four floating point attributes correspond to:

sepal length in cm 2. sepal width in cm 3. petal length in cm 4. petal width in cm The string attribute is the iris class, one of the following: -- Iris Setosa -- Iris Versicolour -- Iris Virginica

What will the program do

1. Read the data from the file. Use only the floating point values for the clustering. Don't discard the class information. We will need it later for assigning names to the clusters and for checking the accuracy of the clusters.
2. Apply the k-means algorithm to find clusters. (There are 3 natural clusters in the case of the iris data.) (See below for more information on k-means.) Use Euclidean distance as your distance measure.
3. Assign each final cluster a name by choosing the most frequently occurring class label of the examples in the cluster.
4. Find the number of data points that were put in clusters in which they didn't belong (based on having a different class label than the cluster name). k-means algorithm:

For each point, place it in the cluster whose current centroid it is nearest and after all points are assigned, update the locations of centroids of the k clusters Repeat for the specified number of iterations.

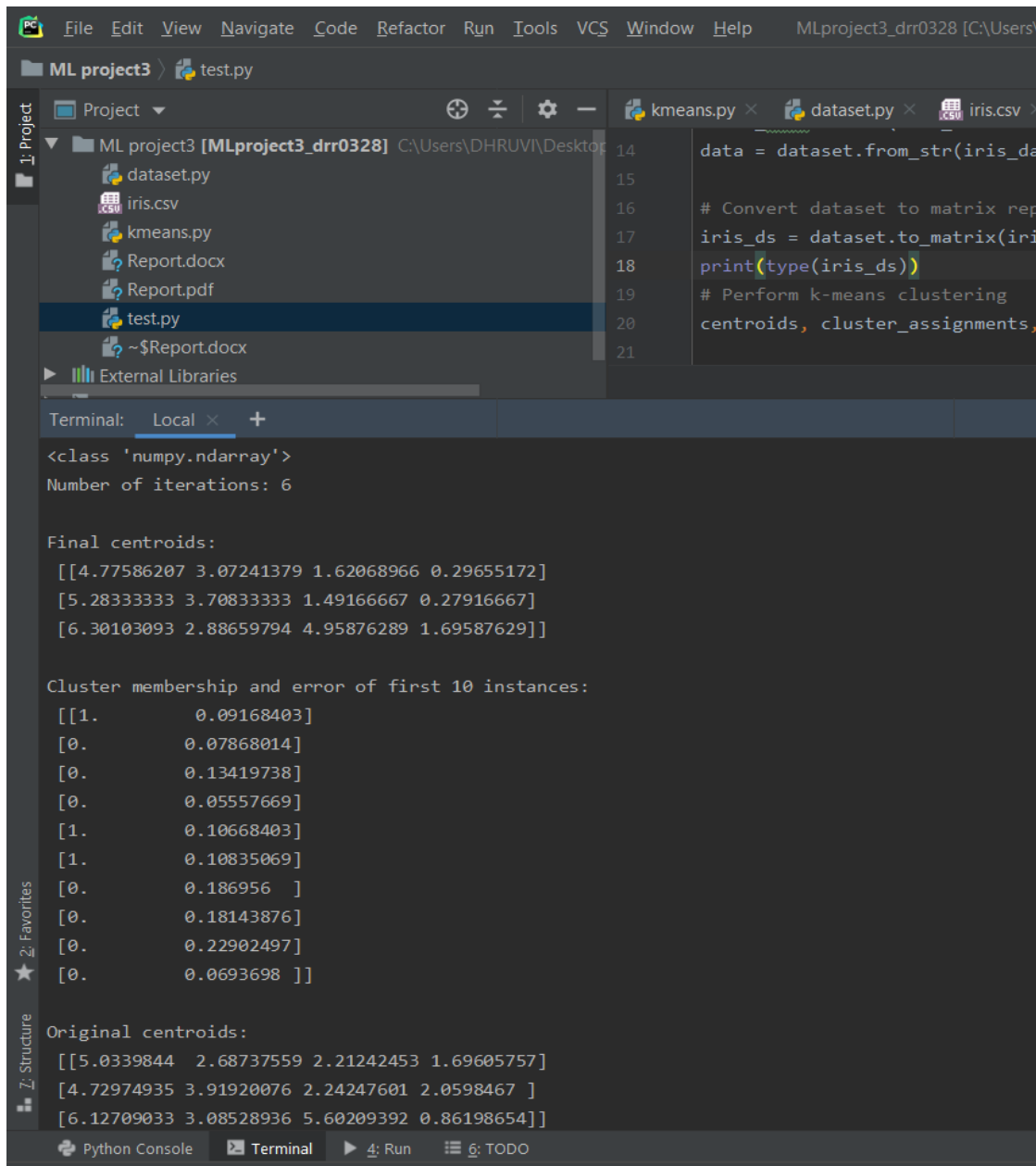
NAME: Dhruvi Rajput

ID: 1001670328

Output of your program

The program will produce output of the form:

Cluster : (List of points in that cluster, one per line) Cluster : (List of points in that cluster, one per line) Cluster : (List of points in that cluster, one per line) Number of points assigned to wrong cluster: (number of points)



```
data = dataset.from_str(iris_da
14
15
16 # Convert dataset to matrix rep
17 iris_ds = dataset.to_matrix(iri
18 print(type(iris_ds))
19 # Perform k-means clustering
20 centroids, cluster_assignments,
21

<class 'numpy.ndarray'>
Number of iterations: 6

Final centroids:
[[4.77586207 3.07241379 1.62068966 0.29655172]
 [5.28333333 3.70833333 1.49166667 0.27916667]
 [6.30103093 2.88659794 4.95876289 1.69587629]]

Cluster membership and error of first 10 instances:
[[1.      0.09168403]
 [0.      0.07868014]
 [0.      0.13419738]
 [0.      0.05557669]
 [1.      0.10668403]
 [1.      0.10835069]
 [0.      0.186956  ]
 [0.      0.18143876]
 [0.      0.22902497]
 [0.      0.0693698  ]]

Original centroids:
[[5.0339844  2.68737559 2.21242453 1.69605757]
 [4.72974935 3.91920076 2.24247601 2.0598467  ]
 [6.12709033 3.08528936 5.60209392 0.86198654]]
```

NAME: Dhruvi Rajput

ID: 1001670328

Analysis phase:

For the number of clusters I have used a number of values for N but the best results can be generated when I considered $N=3$.

For $N=5$ and $N=7$ we get the results showing a cluster centroid which has 'Nan' values. Thus no centroid like that exists and it's not possible.

Also, we know there are 3 total clusters in the iris.data and we know each cluster has exactly 50 data points, and we get similar results in $N=3$.