

## 1) Explanation of The Souce Code

k-Means is often used to pre-cluster massive data sets before other processing. Although it is very simple and fast it is very sensitive to 2 items which are the initial choice of centroid and the Outliers. It is designed to be used on continuous data.

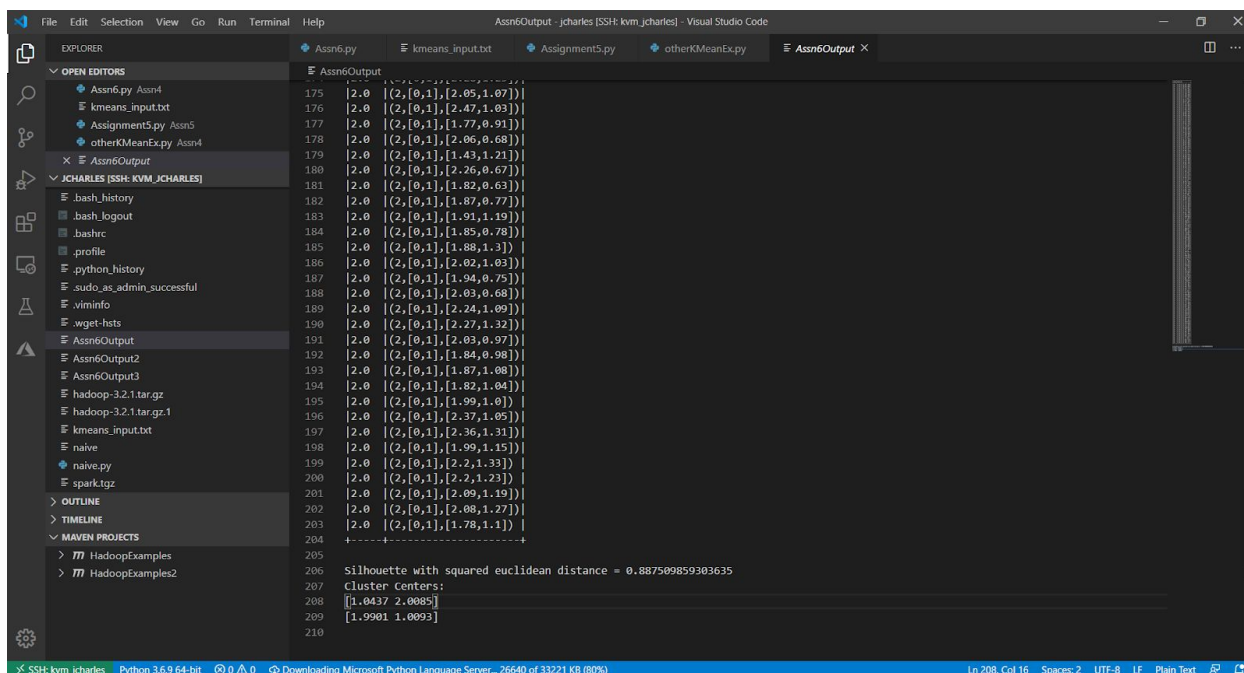
This gif basically explains what the k-means algorithm does, and it is how I perceived it.

[https://thumbs.gfycat.com/NecessarySociableCentipede-size\\_restricted.gif](https://thumbs.gfycat.com/NecessarySociableCentipede-size_restricted.gif)

To Start, we need some additional import like our Kmean import and the ClusteringEvaluator import. These help use their functions later. We start by reading in our dataset with our pregven file by the professor and show our 200 point data set. Afterwards we set the number of clusters, our means, to 2. Following that we then fit our dataset into the model and transform it so our CulteringEvaluator can evaluate it. This gives us the silhouette with sq'd Euclidean distance. Finally we can then get the cluster centers by calling clusterCenters() function on our model. Then we can print.

## 2) Experimental Results:

Issues that I had to deal with was getting my numpy installed and getting the starter code from Spark (which was pretty helpful) , as well as looking for the proper commands.



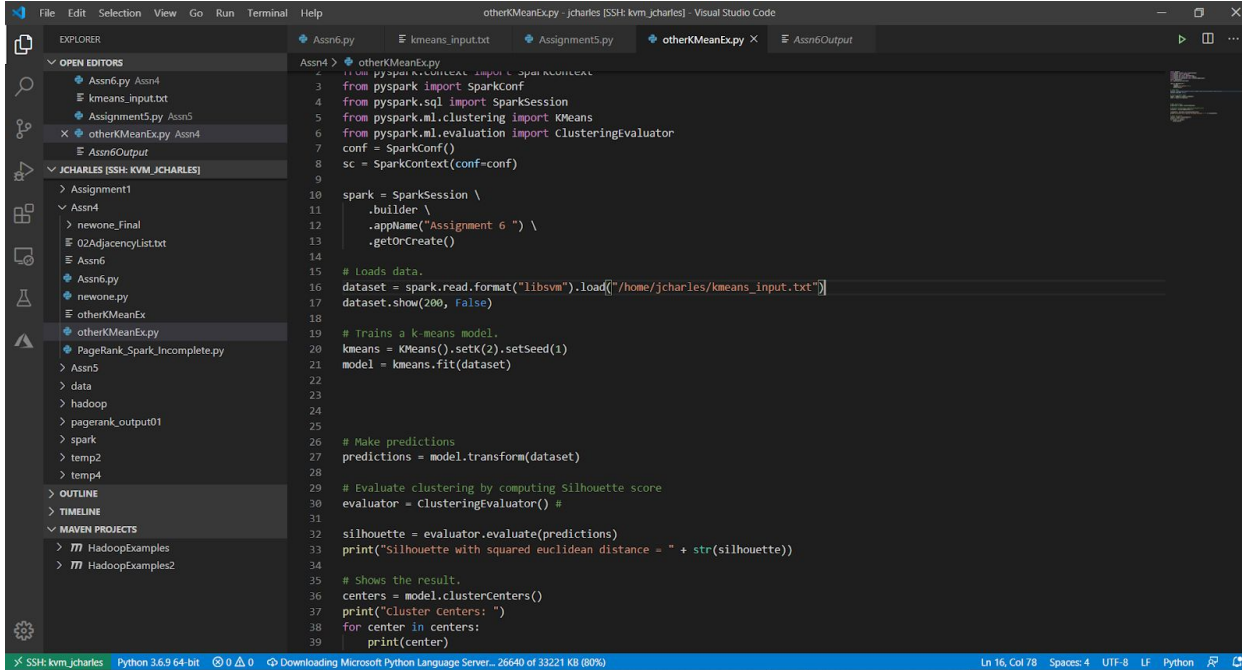
The screenshot shows a Visual Studio Code editor window with a Jupyter Notebook open. The notebook contains a series of cells, with the last cell displaying the results of a k-means clustering analysis. The results include a silhouette score and cluster centers.

```
175 [2.0 [2.05, 1.07]]
176 [2.0 [2.47, 1.03]]
177 [2.0 [1.77, 0.91]]
178 [2.0 [2.06, 0.68]]
179 [2.0 [1.43, 1.21]]
180 [2.0 [2.26, 0.67]]
181 [2.0 [1.82, 0.63]]
182 [2.0 [1.87, 0.77]]
183 [2.0 [1.91, 1.19]]
184 [2.0 [1.85, 0.78]]
185 [2.0 [1.88, 1.31]]
186 [2.0 [2.02, 1.03]]
187 [2.0 [1.94, 0.75]]
188 [2.0 [2.03, 0.68]]
189 [2.0 [2.24, 1.09]]
190 [2.0 [2.27, 1.32]]
191 [2.0 [2.03, 0.97]]
192 [2.0 [1.84, 0.98]]
193 [2.0 [1.87, 1.08]]
194 [2.0 [1.82, 1.04]]
195 [2.0 [1.99, 1.0]]
196 [2.0 [2.37, 1.05]]
197 [2.0 [2.36, 1.31]]
198 [2.0 [1.99, 1.15]]
199 [2.0 [2.2, 1.33]]
200 [2.0 [2.2, 1.23]]
201 [2.0 [2.09, 1.19]]
202 [2.0 [2.08, 1.27]]
203 [2.0 [1.78, 1.1]]
204 +-----+
205
206 Silhouette with squared euclidean distance = 0.887509859303635
207 Cluster Centers:
208 [1.0437 2.0085]
209 [1.9901 1.0093]
210
```

The status bar at the bottom indicates the file is 'Ln 208, Col 16' and the text is 'UTF-8 LF Plain Text'.

I'm also just going to include my source code in the document as well to avoid any confusion with submission.

My source code has comments explaining what each portion of my code does. Although it was short, the simplicity was very appreciated.



```
otherKMeanEx.py - jcharles [SSH: kvm_jcharles] - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  OPEN EDITORS
    Assn6.py Assn4
    kmeans_input.txt
    Assignment5.py Assn5
    otherKMeanEx.py Assn4
    Assn6Output
  JCHARLES [SSH: KVM_JCHARLES]
    Assignment1
    Assn4
    newone_Final
    02AdjacencyList.txt
    Assn6
    Assn6.py
    newone.py
    otherKMeanEx
    otherKMeanEx.py
    PageRank_Spark_Incomplete.py
    Assn5
    data
    hadoop
    pagerank_output01
    spark
    temp2
    temp4
    OUTLINE
    TIMELINE
  MAVEN PROJECTS
    HadoopExamples
    HadoopExamples2
Assn4 > otherKMeanEx.py
1 from pyspark.conf import SparkConf
2 from pyspark.sql import SparkSession
3 from pyspark.ml.clustering import KMeans
4 from pyspark.ml.evaluation import ClusteringEvaluator
5 conf = SparkConf()
6 sc = SparkContext(conf=conf)
7
8
9
10 spark = SparkSession \
11     .builder \
12     .appName("Assignment 6 ") \
13     .getOrCreate()
14
15 # Loads data.
16 dataset = spark.read.format("libsvm").load("/home/jcharles/kmeans_input.txt")
17 dataset.show(200, false)
18
19 # Trains a k means model.
20 kmeans = KMeans().setK(2).setSeed(1)
21 model = kmeans.fit(dataset)
22
23
24
25
26 # Make predictions
27 predictions = model.transform(dataset)
28
29 # Evaluate clustering by computing silhouette score
30 evaluator = ClusteringEvaluator() #
31
32 silhouette = evaluator.evaluate(predictions)
33 print("Silhouette with squared euclidean distance = " + str(silhouette))
34
35 # Shows the result.
36 centers = model.clusterCenters()
37 print("Cluster centers: ")
38 for center in centers:
39     print(center)
```