Clustering

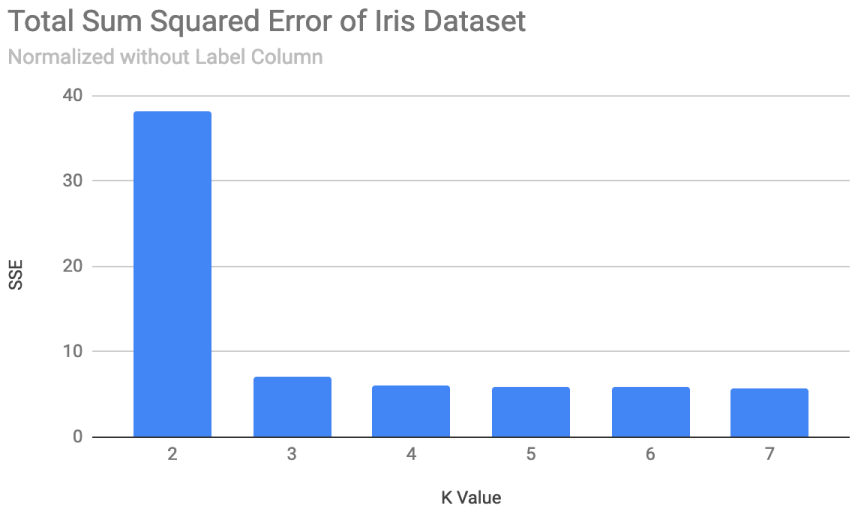
1. I correctly implemented the K Means Algorithm with an arbitrary value k as the number of clusters and that runs iterations until hitting the stopping criteria which is a threshold change in SSE. My code can be found on Learning Suite under the code submission. The file name is cluster.py

   I ran the K Means Algorithm on the Sponge dataset and the results and description are found below.

Mean	SSE		8th Iteration Clusters On Sponge Data						
Centroid 1	51.166	1capa	sincapainternadelcortex	si	2.444	sintilostilosadicionales	sinespiculaprincipalestilo'	2.611	0.444
Centroid 2	27.000	sincortex	sincapainternadelcortex	no	0.000	sintilostilosadicionales	sinespiculaprincipalestilo'	2.000	0.000
Centroid 3	5.750	sincortex	sincapainternadelcortex	no	0.000	sintilostilosadicionales	sinespiculaprincipalestilo'	0.500	1.250
Centroid 4	80.440	2capas	tangencial'	si'	3.040	intermediarios'	sinespiculaprincipalestilo'	2.520	2.480
Tot. SSE	164.356								

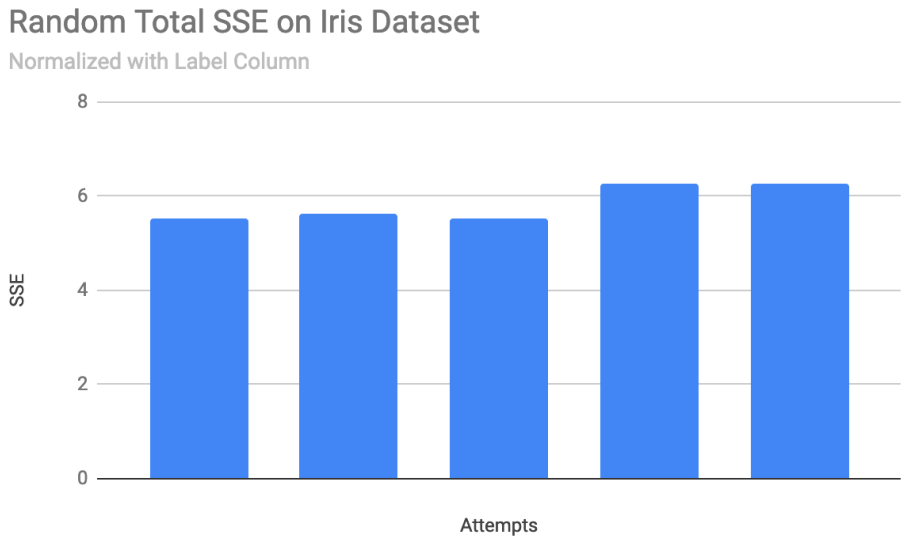
This table of the centroids after running K Means algorithm on the Sponge dataset with k = 4. The algorithm ran 8 iterations before stopping. Notice that the Total Sum Squared Error (SSE) can be found at the bottom of the graph at a value of 164.356. The individual SSE of each centroid is found on the left most column and the values at each centroid are found next to the id Centroid #.

2. I used the K Means Algorithm on the Iris dataset with a the value k from 2-7. And the results for are shown in the graph below as well as a description.



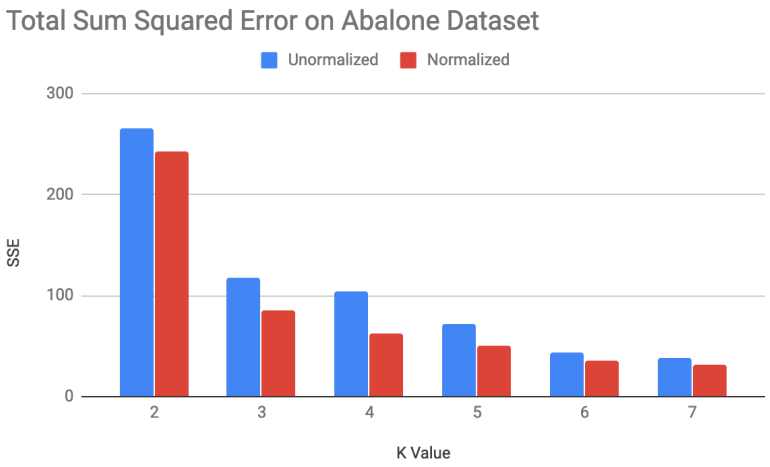
This graph show the Total Sum Squared Error (SSE) on the y-axis and the k values from 2 to 7 on the x-axis. Notice how much of a decrease some when there is more than two centroid. After shifting from 2 to 3, the value decreases only slightly. This can be specific to the dataset and a general rule that after a certain number of centroids, the SSE doesn't drastically improve.

I then used the K Means Algorithm on the Iris dataset with a the value k = 4 and random centroids. And the results for are shown in the graph below as well as a description.



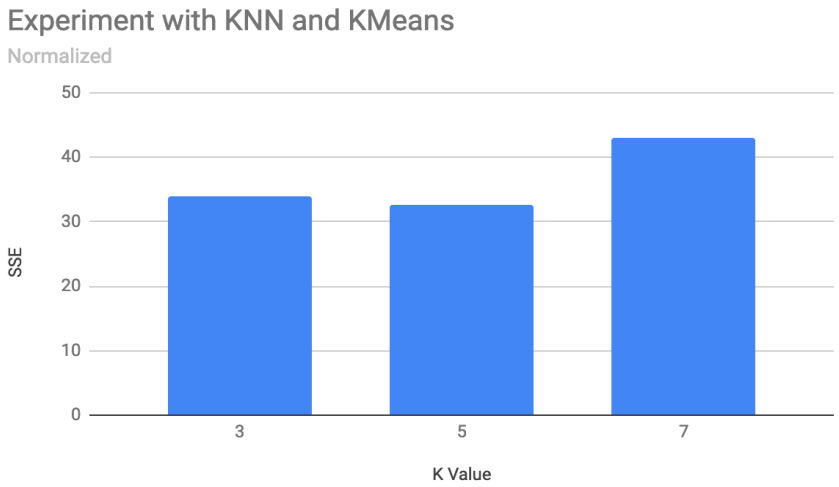
This graph show the Total Sum Squared Error (SSE) on the y-axis and the k = 4 attempts on the x-axis. These attempts have randomly selected initial centroids and includes the output layer. Notice how the value doesn't really change. It seems that there might be an increase, but that doesn't not mean it will always increase. Adding the output layer didn't change my k = 4 values are not too different, the SSE decreased slightly.

3. I ran K Means on the Abalone Dataset including the attribute 'rings' and treated it as a continuous value because the values range from 1 to 29 and those values are not ID's they are an actual integer of the number of rings and therefore the value matters, not just the index. The results of the SSE for each of the k values from 2 to 7 are shown below as well as a description.



This graph shows of the Sum Square Error of the Abalone dataset with and without normalizing the data. The normalized data is in red and the not normalized data is in blue. Note that they both decrease with time, but the normalized data has a smaller Sum Squared Error.

5. For my experiment I used the K Nearest Neighbor Algorithm and the K Means Algorithm to find first the new centroids and then use KNN and only keep the values that are within a specific distance of that mean and then use only those values to calculate the new centroid and set the rest to a missing value of ?. Below is the Graph showing the SSE with k = 2 though k = 7 for the this new combination of KNN and K Means for the Iris Data set. See code at kmeans_knn.py



This table of the Centroids after running K Means algorithm on the Sponge dataset with k = 4. The algorithm ran 8 iterations before stopping. Notice that the Total Sum Squared Error (SSE) can be found at the bottom of the graph at a value of 164.356. The individual SSE of each centroid is found on the left most column and the values at each centroid are found next to the id Centroid #.