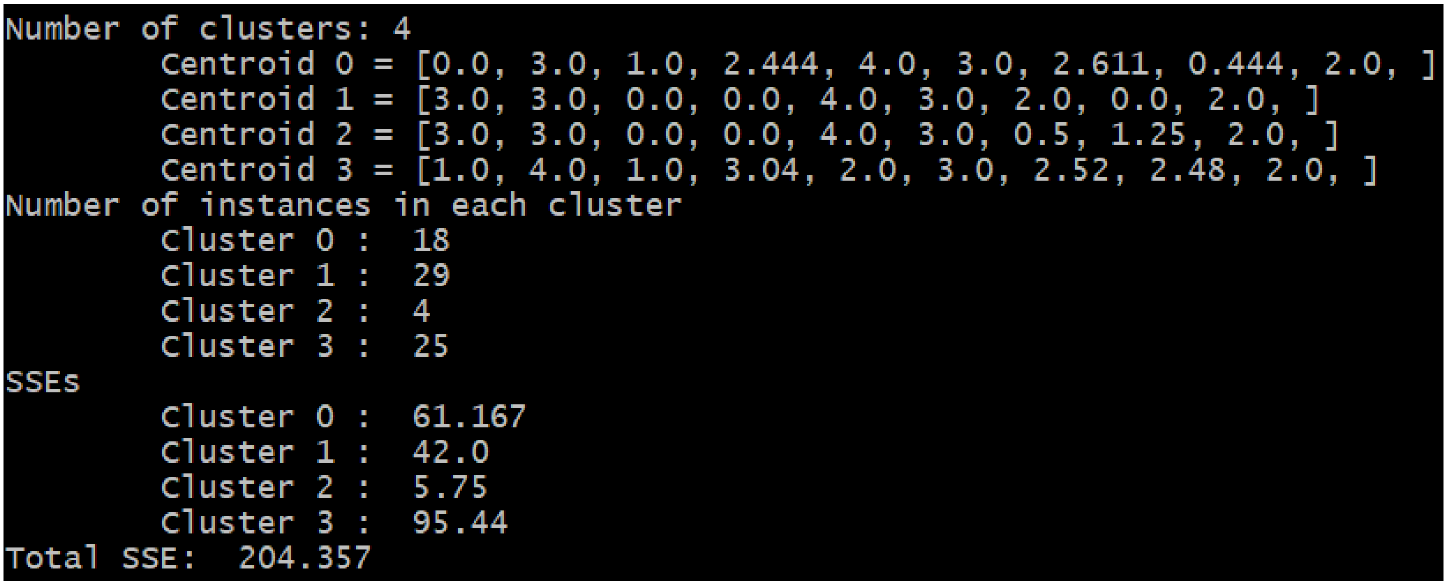
Paul Johnston

CS 478

Clustering

1. This clustering model uses k-means to cluster given data. It calculates the total SSE of the clusters after every iteration and stops changing the clusters after the SSE converges. When calculating distance between two instances of data or from a cluster centroid to any given instance, it counts matching nominal values as a distance of zero, mismatching nominal values as a distance of 1, any unknown values as a distance of 1, and for real values it takes the difference. After taking these differences it calculates Euclidean distance on these differences. K can be any positive value and it provides the option of using the output label and first column as feature data as well as choosing random k instances as the initial centroids or the first k instances.

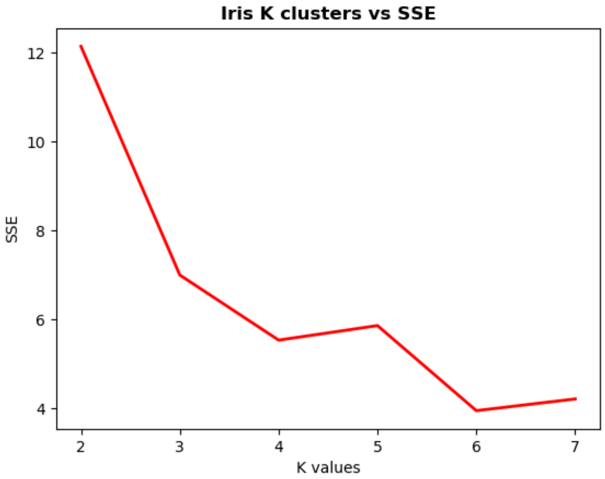
The following output was produced after running the model on the sponge dataset with all the columns as features and no normalizing. K was 4 and the first 4 instances of data were used as the initial k centroids.



```
Number of clusters: 4
        Centroid 0 = [0.0, 3.0, 1.0, 2.444, 4.0, 3.0, 2.611, 0.444, 2.0, ]
        Centroid 1 = [3.0, 3.0, 0.0, 0.0, 4.0, 3.0, 2.0, 0.0, 2.0, ]
        Centroid 2 = [3.0, 3.0, 0.0, 0.0, 4.0, 3.0, 0.5, 1.25, 2.0, ]
        Centroid 3 = [1.0, 4.0, 1.0, 3.04, 2.0, 3.0, 2.52, 2.48, 2.0, ]
Number of instances in each cluster
        Cluster 0 :  18
        Cluster 1 :  29
        Cluster 2 :  4
        Cluster 3 :  25
SSEs
        Cluster 0 :  61.167
        Cluster 1 :  42.0
        Cluster 2 :  5.75
        Cluster 3 :  95.44
Total SSE:  204.357
```

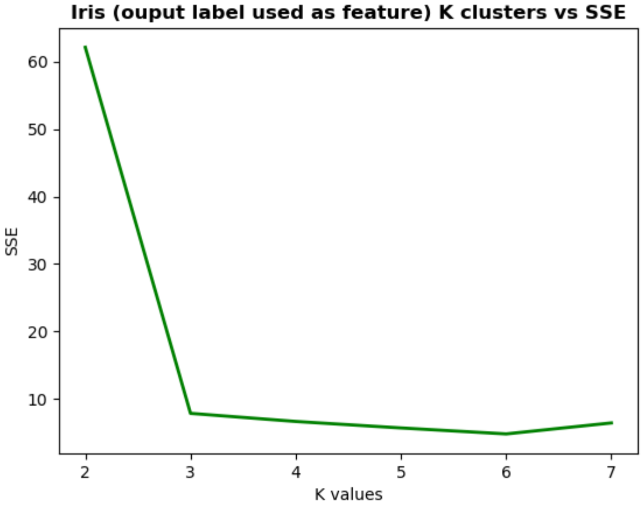
2. Iris

The following output was produced when running the model on the full iris dataset where the output label was not included as part of the features. If there were ever any empty clusters, the model was re-run with different initial k centroids. K random points were chosen as the initial centroids and the data was normalized. Because Euclidean distance was used, normalization helped the model to look at all the attributes instead of just the largest valued ones.



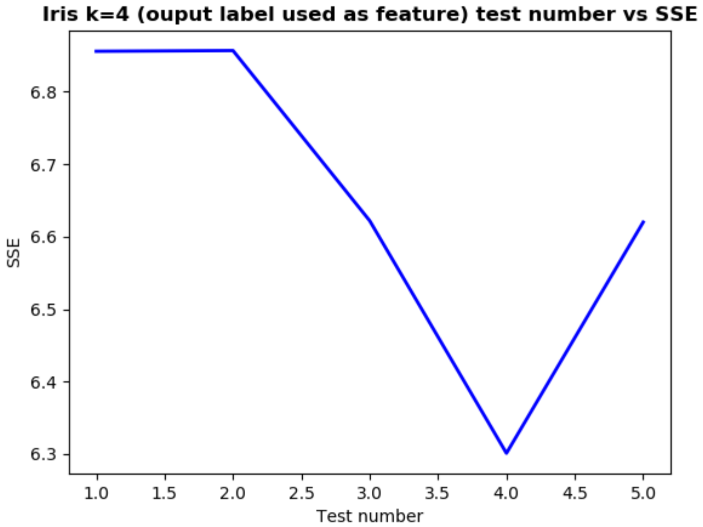
As k increases, the clusters are becoming more and more similar to each other. This is shown by the centroid values. As k increases, the centroid values become more similar to each other, and as the centroid values are representations of the clusters, it can be inferred that the clusters are getting closer and closer to each other. The dramatic decrease in SSE at 3 clusters makes sense because there are kinds of iris flowers, however, because the output label is not being used there is some fluctuation when k is greater than three.

The following data was collected after running the clustering model on the iris dataset with normalization and the output label being used as an input feature. Random initial k centroids were used



Since the data was attained from using the output label, the SSE dropped dramatically at k=3 because the output label was a telling factor in deciding the clusters. With k > 3, the sse did not change much because the output label helped classify the clusters and not make them so spread out. I noticed that when using k=5, the features with the same output labels were still clustered together, it’s just that there were multiple clusters per class and the centroid values were very similar.

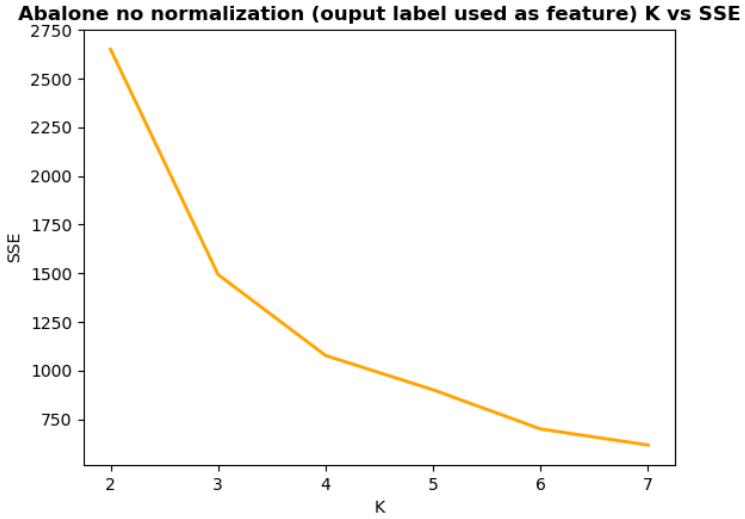
The following data was collected after running k-means 5 times with k=4, each time using different random initial k centroids.



There is some slight variation in the SSEs. Because k random initial centroids were chosen, the clusters were never exactly the same, even after many iterations of the algorithm. The centroids will get closer and closer to the "true" values but never exactly the same unless the same k centroids were chosen twice. The difference between test 1 and 2 is 0.001, so they were not exactly the same initial k centroids. The variation overall is 0.556 which is not very much compared to the variation when using different k values which was ~60.

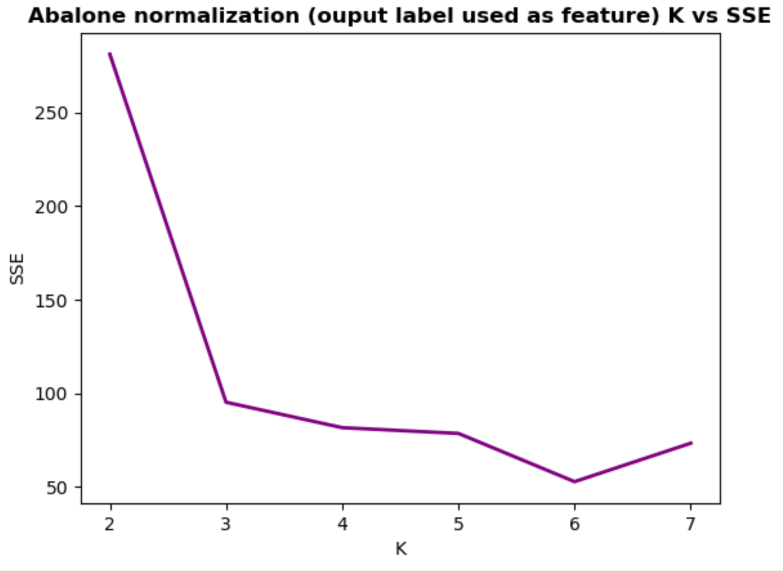
3. Abalone

The following data was collected from running k-means on the abalone dataset without normalization, using the output label as an input feature and treating the output label "rings" as continuous. The output label is being treated as a continuous variable because it allows for better generalization and normalization of the label in order to have the distance metric be more informative than a nominal difference would of 1 or 0.



The SSEs are very high. When not normalizing, the distance is greater in general because features such as "rings" whose values are generally high pull the SSE upwards because the distance from instance to centroid is greatly affected by this difference. It's interesting to note that the SSE decreases gradually and has only a sudden drop from 2 to 3 clusters but from then on, slowly decreases.

The following data was a repeat of the above data with normalization.



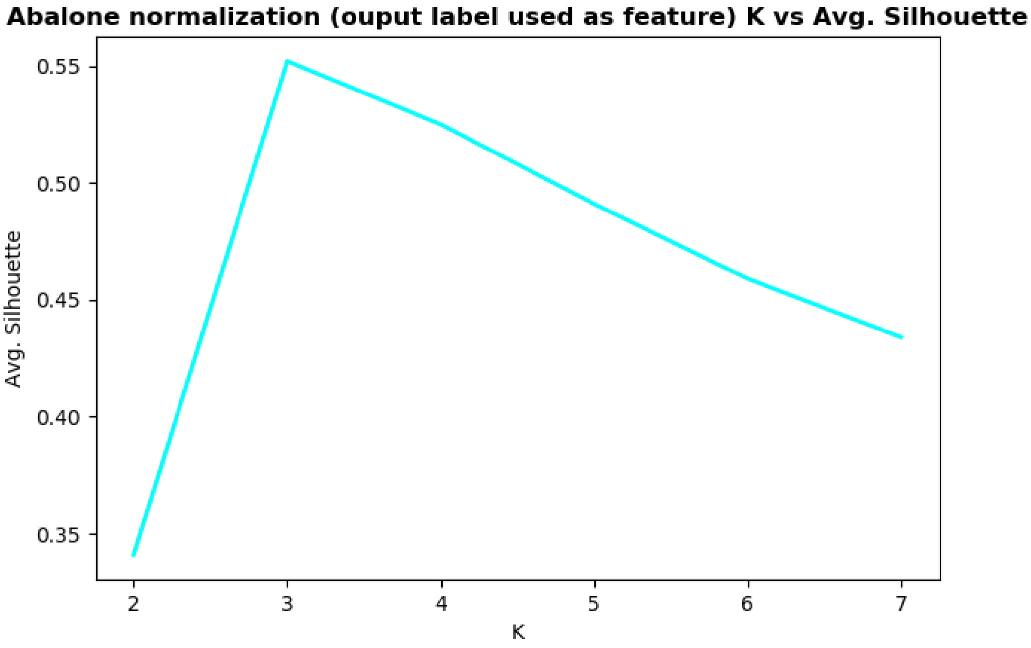
It’s interesting to note that the sse decreased until k was equal to 7 even though there are more than 7 output label values present in the data. I attribute this to two or more centroids being chosen that were relatively close together and the others were spread out over many leftover points, resulting in a greater SSE. The main difference between this data from normalization and the data from not normalizing is that the SSEs are much smaller for the normalized set. This is because large values found in “rings” are brought between 0 and 1, making their effect on overall distances between centroids and instances less.

4. Silhouette

The following data was collected after running k-means on the abalone dataset using normalization and k values from 2 to 7. The data is focused on silhouette scores which were calculated in the “calcSilhouetteScore” in the clustering learner.

		Individual cluster silhouette scores							Average Silhouette
		1	2	3	4	5	6	7	
K	2	0.243	0.52						0.341
	3	0.485	0.624	0.556					0.552
	4	0.514	0.408	0.545	0.599				0.525
	5	0.531	0.425	0.474	0.509	0.518			0.491
	6	0.365	0.432	0.424	0.453	0.434	0.525		0.459
	7	0.281	0.44	0.24	0.42	0.316	0.234	0.561	0.434

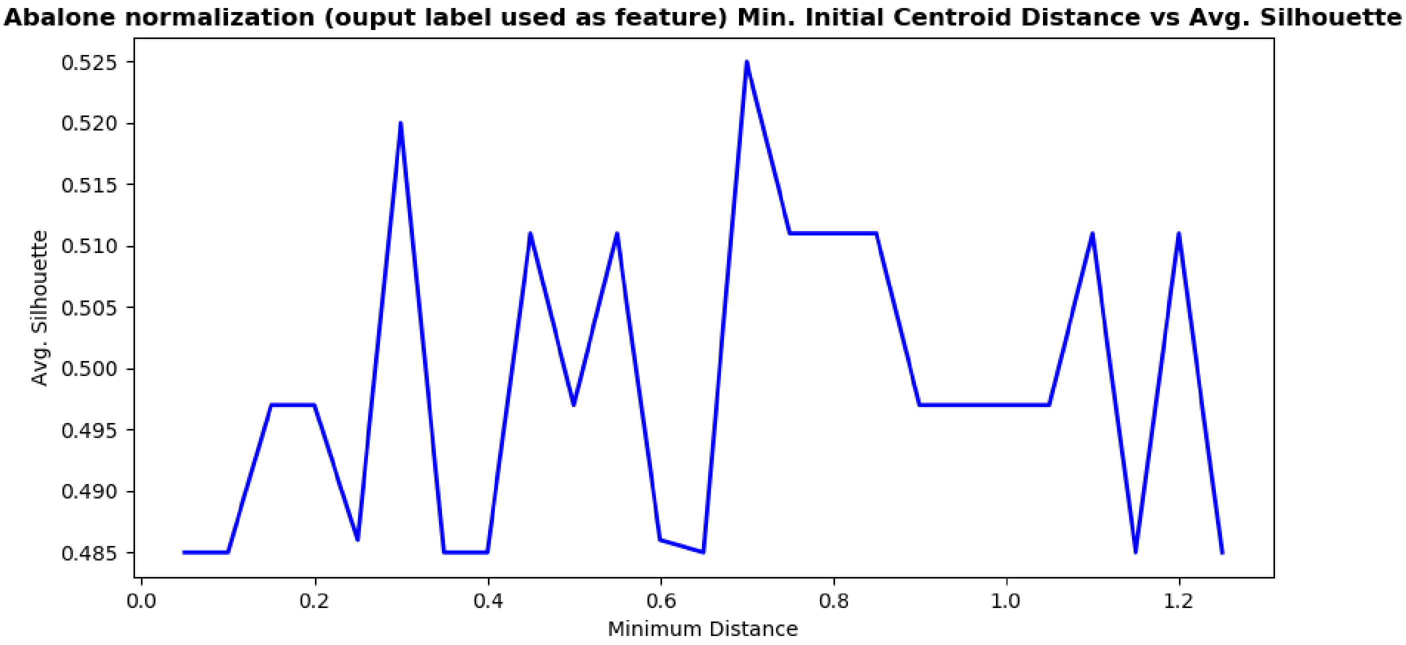
The following graph shows the k values vs the average silhouette scores for that clustering.



The silhouette score was the highest when k was equal to 3. This agrees with the results found when measuring the clusters' accuracy with the SSE. The k value that resulted in one of the lowest SSEs was 3, the lowest being 6. The silhouette scores slightly, very slightly, decreased as k increased past 3. I attribute this slight variation between the silhouette scores and SSEs to the random initial k centroids chosen. This could have slightly altered one result of k for either test but it is interesting to note that spikes in both graphs (dramatic decrease in SSE, dramatic increase for silhouette score) both occur when k=3.

5. Experiment

For my experiment, I tried to spread out the initial k centroids as much as I could, to varying degrees, choosing different minimum distances that the initial centroids had to be from each other, to see if that improved overall silhouette scores. I ran the tests on the abalone dataset using normalization and the output label as an input feature and k = 4 The graph below shows minimum distance required vs average silhouette score.



What I found through this data was that the minimum distance restraint didn’t so much improve the overall clustering as the minimum distance increased as it did prevent the clustering from being horribly bunched. I printed out how many instances it went through before choosing the k initial centroids that were far enough apart and the most it got to was 237. I think that this method only stops the exceptionally bad initial clusterings from happening, but doesn’t lead to overall improvement as the minimum distance increased as you can see with the spikey variance.