Lindsay Sheppard

DSC 609: Machine Learning

Programming Project #8: K-Means Clustering

**K-Means Clustering Overview**

K-means clustering algorithms are simple, robust, and popular unsupervised learning methods that attempt to group, or cluster, similar objects together without supervision in the form of labeled data. First described as ‘k-means’ by MacQueen (1967), they are useful for numeric, continuous data with smaller dimensions. They are often used for exploratory data analysis and have been successful in document classification, high-crime area identification, market segmentation, fraud detection, and cyber-profiling, to name a few (Raghupathi, 2018).

K-means attempts to split data into a fixed number (k) of clusters. The optimal choice of k strikes a balance between maximum compression of the data with a single cluster and maximum accuracy by assigning each data point to its own cluster (Determining, n.d.). K is chosen by the user, either at random or through the examination of elbow plots or certain indices, such as the Adjusted Rand (AR) and Calinski-Harabasz (CH). For example, one may choose k=3, which randomly initializes three cluster centroids, or data points in the center of each cluster. The choice of three cluster centroids groups the data into three clusters. The algorithm is iterative and minimizes the Euclidean distances between every data point and its centroid, while maximizing the distance between the cluster centroids (Landman, Pang, Williams, & Ross, 2019). It performs two steps: cluster assignment and centroid relocation (Jeevan, 2017). During the first step, the algorithm examines each data point and assigns the data points to one of the three cluster centroids based on which cluster is closer. The second step calculates the average for all cluster points and relocates the cluster centroid to that location. The process iterates, using different random centroids each time until there is basically no change in the clusters.

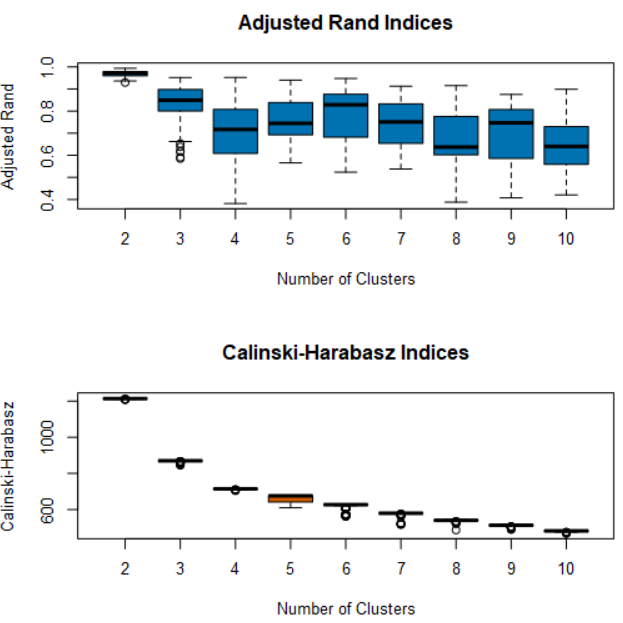
**Data Preparation**

Alteryx Designer version 2018.3.5.52487 uploaded the file Assign4\_5\_6\_7\_8\_Data\_File.csv using the Input Data tool. The file contained the following twelve numeric attributes: *fixed\_acidity, volatile\_acidity, citric\_acid, residual\_sugar, chlorides, free\_sulfur\_dioxide, total\_sulfur\_dioxide, density, pH, sulphates, alcohol,* and *quality.* The .csv file format uploads all columns as strings; all variables were changed to doubles using the Select tool. The dataset had been used previously to examine other types of algorithms and was known to not contain any missing data.

The K-Centroids Diagnostics tool assisted in determining *k*, the number of clusters to specify for the clustering analysis subsequently performed by the K-Centroids Cluster Analysis tool. The former tool provided a guide for how the data naturally divided based on starting input from the user. The configuration menu of the Diagnostics tool offered standardization using z-score or unit interval; as clustering algorithms are very sensitive to the scale of the input data, z-score standardization was chosen for the current report (Firman [1], 2019). It also provided three options of clustering: K-Means, K-Medians, and Neural Gas; as it is the primary focus of this report, K-means was chosen.

The minimum number of clusters defaulted to 2 and the maximum was chosen to be 8, though the tool allowed for up to 70. The “Bootstrap replicates” option determined how many random samples with replacement were created and consequently, how many times each clustering solution was repeated (Firman, 2019). The minimum is 50 and the maximum is 200; 100 was selected as a starting point. A higher number may result in better accuracy but is increasingly computationally expensive. The number of starting seeds (i.e. initial centroids) may be set between 1 and 10; 5 was chosen for the current report.

Two measures evaluated the performance of the initially chosen inputs to determine a potentially optimal k value. The AR index measured how similar two clustering solutions are; on a scale of 0 to 1, a value of 0 suggests totally random clusters (Firman [1], 2019). The CH index measured how close the data is within the clusters as well as how separated the clusters are. It is generally more accurate when the clusters are approximately spherical in shape and similar in size. With both indices, a higher value is indicative of a better solution. Figure 1 displays the Diagnostic tool results.

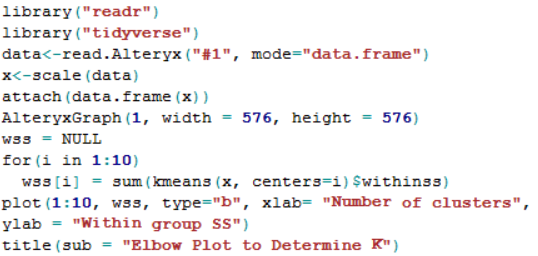


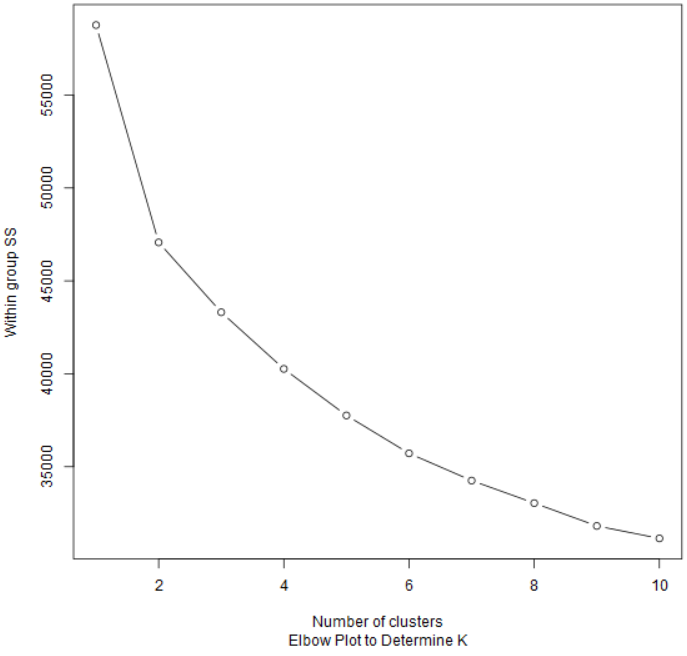
**Figure 1:** K-Centroids Diagnostic Tool Output (k= 2-10, number of starting seeds = 5)

The determination of the ‘best’ number of clusters is subjective. Based on this output, it appeared that when k=2, the AR and CH indices both reported the highest median, as shown by the black horizontal line in each boxplot.

Another way to approach k selection is to examine an elbow plot of within-sum-of-squares (WSS) values (y-axis) versus potential values of k (x-axis). Where the plot bends is a decent approximator of the optimal k-value, because the clustering algorithm attempts to define clusters so that the total intra-cluster variation (i.e. total within-cluster variation) is minimized (K-means, n.d.). The total within-cluster sum of square measures how compact clusters are; more compactness is preferred.

The best way to create an elbow plot, or any plot for that matter, is by using either the R or Python tool and writing code to perform the desired task. Figure 2 shows code adapted from Reddy (n.d.) and the elbow plot that it created. Again, choosing the ‘bend’ of the plot is subjective. In the resulting plot, k ranging from 2-5 may be chosen, depending on where a user defines the plot’s elbow, with k=2 the most likely choice.

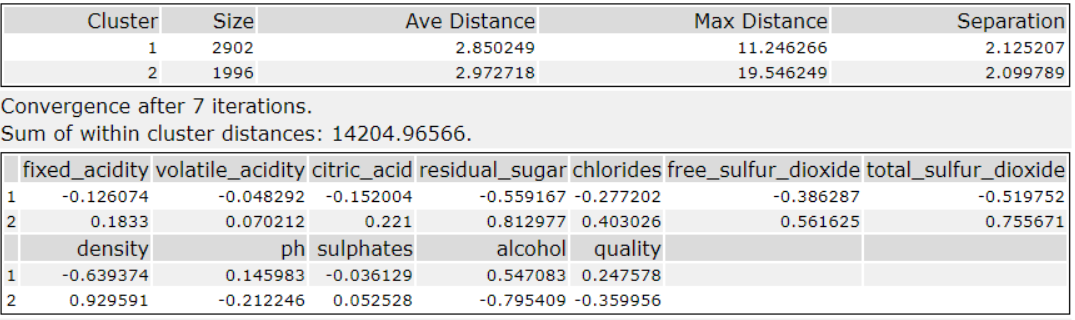




**Figure 2:** Elbow plot generated by the R tool

**Cluster Analysis**

For the subsequent cluster analysis performed with the K-Centroids Cluster Analysis tool, the same parameters as the Diagnostics tool were used: k=2 and number of starting seeds=5. The results are shown in Figure 3.

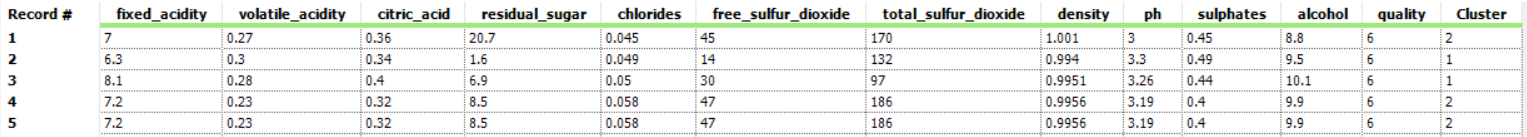


**Figure 3:** K-Centroids Cluster Analysis (k=2): Results

The first table showed the cluster results. The cluster numbers 1 and 2 were arbitrarily assigned. Approximately 59.2% of the data fell into Cluster 1 and 40.7% into Cluster 2. If one cluster had contained a much larger percentage of the data than the other, one might consider attempting different parameters in the Cluster Analysis tool and trying again or examining the values in the smaller cluster to see if they are highly unusual.

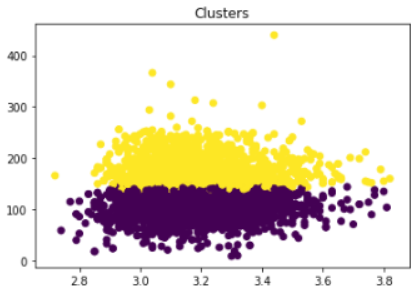
Cluster 1 reported the smallest average distance of 2.85, indicating it is more compact than Cluster 2. Max distance does not provide much information except to show the location of the most outlying data point for each cluster. Separation shows the distance of the closest point not in the relevant cluster. With only two clusters, the separation result was expected to be about the same in each cluster, and it was.

The chart with each variable listed is read within each variable and not horizontally for best results. Values within variables that have large positive and negative values indicate the clusters tend to oppose each other for that variable. Cluster 1 and Cluster 2 reported opposite values in every single column. It is important to note that the positive and negative signs do not indicate largest and smallest values. For example, the negative pH value in Cluster 2 does not necessarily indicate that Cluster 2 had smaller pH values than Cluster 1, but that they are very different from each other.

To see which records were located in each cluster, an Append Cluster tool was used. The tool is useful if a user wanted to add a cluster identifier to each record or output the file to be combined with raw data for further validation. An example of the tool’s output may be seen in Figure 4.

**Figure 4**: Append Cluster output

Finally, the plots generated by the Cluster Analysis tool are not always easily interpretable. Therefore, the R or Python tool may be used to examine the clusters’ spatially relative locations. The Python tool is setup like Jupyter Notebook and contains several common libraries, including jupyter, matplotlib, scikit-learn, numpy, and pandas. Using scikit-learn to generate a quick plot of the clusters, one can examine a scatterplot with desired variables to examine cluster differences. Figure 5 displays a plot generated by the Python tool that visualizes clusters using the attributes ph and total\_sulfur\_dioxide. The plot code was saved within the Alteryx Python tool. One may notice that the yellow cluster tended to have about the same pH levels (x-axis) as the purple cluster but reported a higher range of total sulfur dioxide levels (y-axis).



**Figure 5**: Clusters 1 and 2, ph vs. total\_sulfur\_dioxide, code adapted from Firmin [2], (2019)

**Conclusion**

K-Means clustering is a useful tool for exploring unlabeled data and grouping it by similarity. Alteryx Designer is a well-equipped tool for performing this task. Not only does a user have the option for K-Means, but he or she may compare his or her results to those of the K-Medians or Neural Gas algorithms as well. Additionally, iteration with the Cluster Analysis tool is simple and quick, and the Diagnostics tool helps to narrow down the initial choice of the k parameter with the AR and CH indices provided. The Append Cluster tool makes it easy to assign clusters to records for further validation. Furthermore, the R and Python tool are versatile and easy to use and can generate other plots for insight as needed.

However, potential downsides are that choosing the k value is still quite subjective and in Alteryx, determining the best number of bootstrapping replicates or starting seeds with the Diagnostics tool can be computationally expensive. The results are not easily interpretable without additional information or further examination into the differences between clusters or without additional plots that the Cluster Analysis tool does not provide. The user must take care to ensure that the results are adequate through careful tuning and validation.

**References**

Determining the number of clusters in a data set. (n.d.) In *Wikipedia.* Retrieved fromhttps://en.wikipedia.org/wiki/Determining\_the\_number\_of\_clusters\_in\_a\_data\_set

Firman, S. (2019, March 8). [1]. Tool Mastery| K-Centroids Diagnostics. *Alteryx Knowledge Base.* Retrieved from https://community.alteryx.com/t5/Alteryx-Knowledge-Base/Tool-Mastery-K Centroids Diagnostics/ta-p/302311

Firman, S. (2019, March 8). [2]. Tool Mastery | Python. *Alteryx Knowledge Base.* Retrieved from https://community.alteryx.com/t5/Alteryx-Knowledge-Base/Tool-Mastery-Python/ta-p/197860

Jeevan, M. (2017, August 7). Possibly the simplest way to explain K-means algorithm. *Big Data Made Simple.* Retrieved from https://bigdata-madesimple.com/possibly-the-simplest-way-to-explain k-means-algorithm/

K-means cluster analysis. (n.d.). *UC Business Analytics R Programming Guide.* Retrieved from https://uc r.github.io/kmeans\_clustering

Landman, N., Pang, H., Williams, C., & Ross, E. (2019). k-means clustering. *Brilliant.org*. Retrieved from https://brilliant.org/wiki/k-means-clustering/

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Retrieved from https://projecteuclid.org/download/pdf\_1/euclid.bsmsp/1200512992

Raghupathi, K. (2018, March 27). 10 interesting use cases for the K-means algorithm. *DZone.* Retrieved from https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm

Reddy, S. (n.d.). kmeans clustering: elbow curve, kselction(sic), animation for selecting k value to form clusters. Retrieved from https://rpubs.com/sravanreddy/kmeans\_clustering