Applied Spatial and Temporal Data Analysis

Homework #3 – K means Clustering  
Mohammed Jasam

1. **Data Extraction**

* Wrote a python script to scrape the articles from CNN.com and saved the data in the csv file.
* Removed all the regular expressions to get meaningful data.
* Created a data matrix in which the first row contains all the meaningful words from every article and the first column represents the article names.
* Ran a frequency analyzer to find out the frequency of those words in every article.

1. **Feature Selection**

The above data matrix is a sparse matrix i.e, it has lots of zeroes rather than useful data. So, we do a feature extraction to retain the meaningful attributes and delete the useless attributes. This increases the accuracy of the clustering algorithms drastically. Also, performing normalization on the data values helps in better cluster formation.

1. **Packages & Libraries**

The program has been scripted in Python 3.6 and a list of modules were used to generate the results

* SciPy: Used for mathematical computations
* NumPy: Used for mathematical computations
* SciKit-Learn: Machine learning module for python
* Beautiful-Soup: Used to extract data from website
* tkinter: Used for visualizing the data

1. **Important Terminology**

***Euclidean Distance:***

Euclidean distance is a standard metric for geometrical problems. It is the ordinary distance between two points and can be easily measured with a ruler in two- or three-dimensional space.

***Cosine Distance:***

The documents which are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity.

***Jaccard Distance:***

The Jaccard coefficient, which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects. For text document, the Jaccard coefficient compares the sum weight of shared terms to the sum weight of terms that are present in either of the two document but are not the shared terms.

***Existence Matrix:*** This matrix has 1’s and 0’s and records the presence/absence of a particular attribute.

***Frequency Matrix:*** This matrix has 1’s and 0’s and records the number of presence of a particular attribute.

1. **K-means Clustering**

***The Algorithm:***

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as vary centers of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.  
Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/images/image009.gif,

where https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/images/image011.gif is a chosen distance measure between a data point https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/images/image013.gif and the cluster centre https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/images/image015.gif, is an indicator of the distance of the n data points from their respective cluster centres.

The algorithm is composed of the following steps:

|  |
| --- |
| 1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.* 2. *Assign each object to the group that has the closest centroid.* 3. *When all objects have been assigned, recalculate the positions of the K centroids.* 4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.* |

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The k-means algorithm can be run multiple times to reduce this effect.

***Experimentation:***

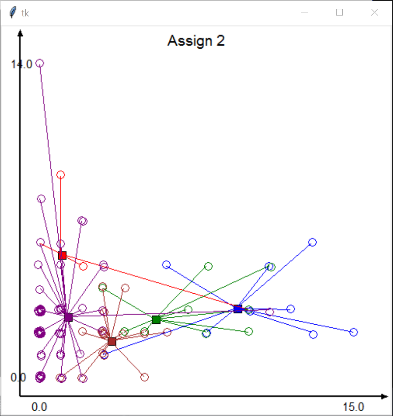
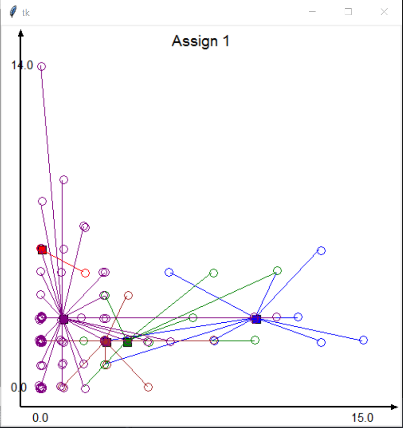
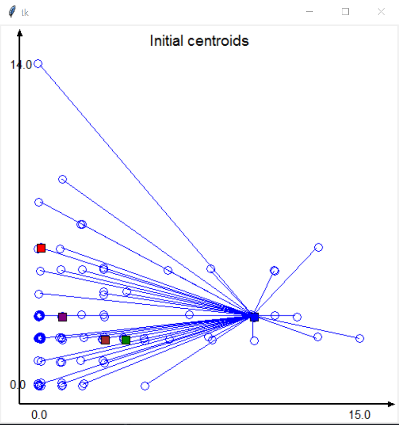
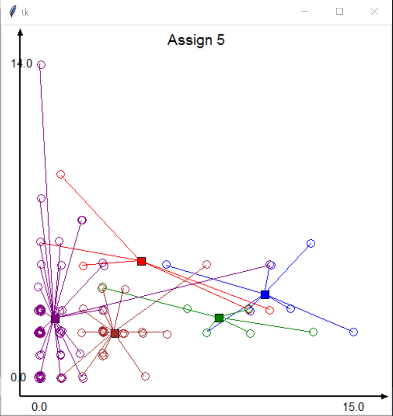
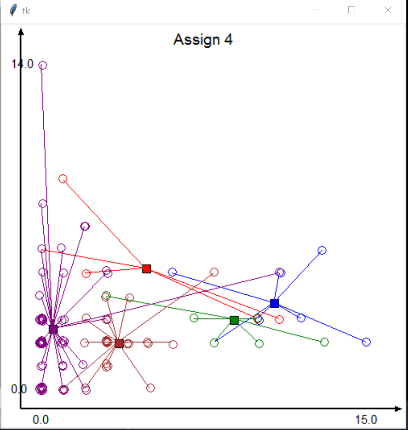
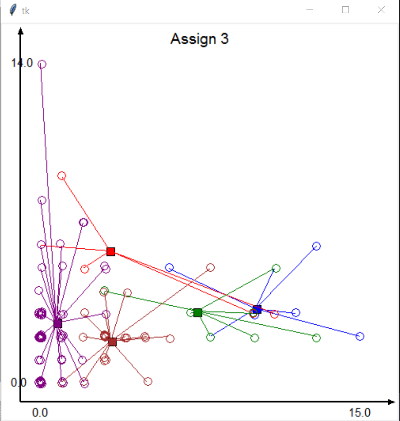
K-means has been initially applied using a distance formula later I have replaced the distance metrics with Euclidean, Cosine and Jaccard Similarity. This gave varying results during cluster formation.

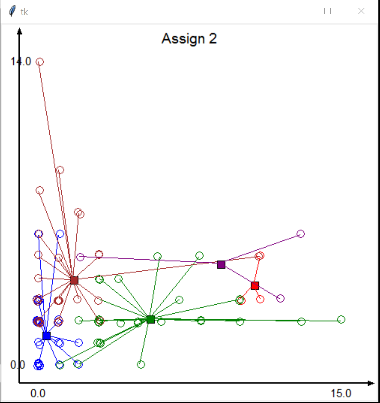
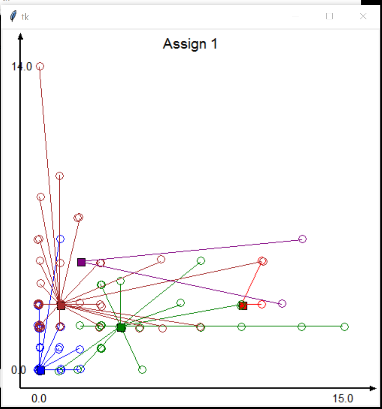
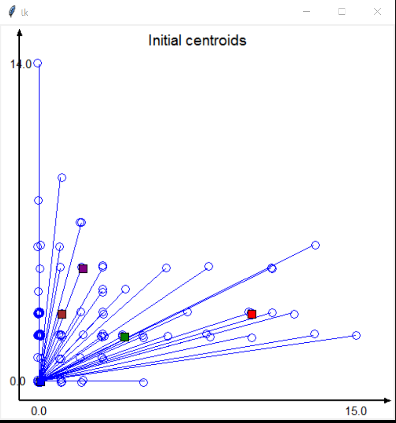
Mostly, Cosine had best results as it formed accurate clusters on Frequency matrix as input. While the Jaccard has varying results which was mainly due to the different input i.e, Existence matrix which only has 1’s and 0’s in its data.

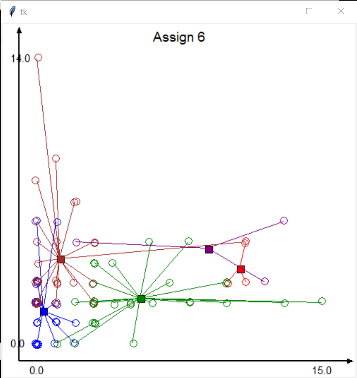
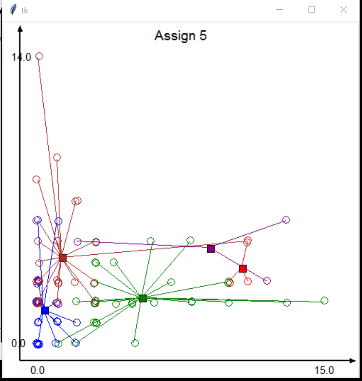
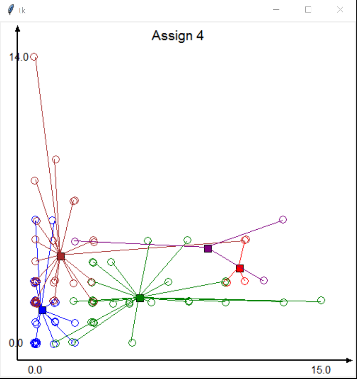
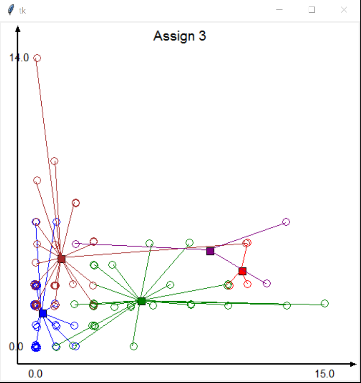
Cluster assignment with Cosine was quicker than all other metrics, moreover, it mainly depends on the selection of the initial k centroids. If the centroids are selected far apart from each other then, the algorithms forms clusters in minimum number of steps and vice versa. All the analysis has been done and data has been visualized using tkinter library in Python.

1. **Visualization**

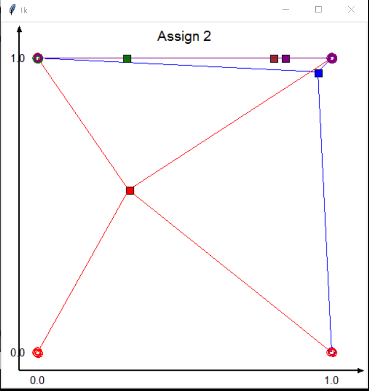
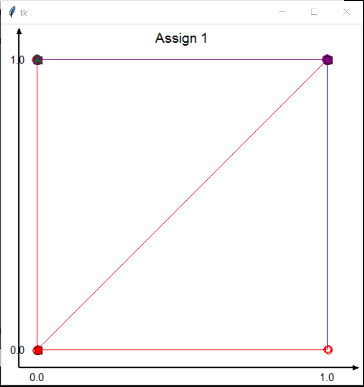
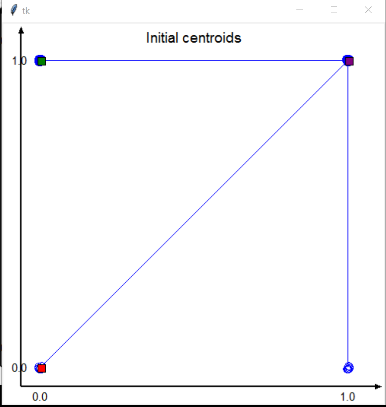
***Cluster Formation using Classic Kmeans:***

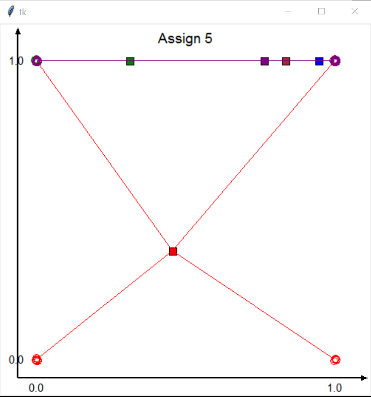
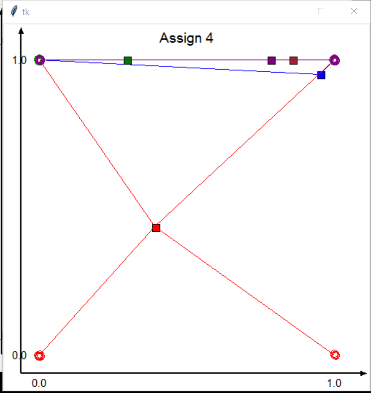
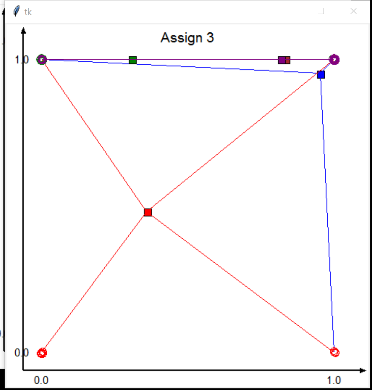
***­***

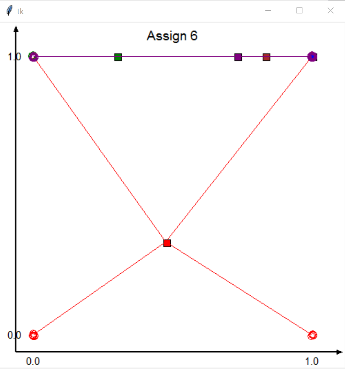
***Cluster Formation using Euclidean Distance as the distance metric: ***

******

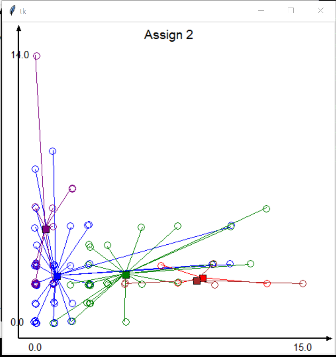
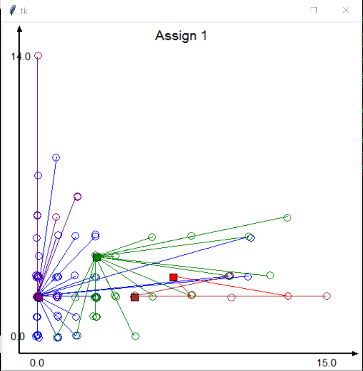
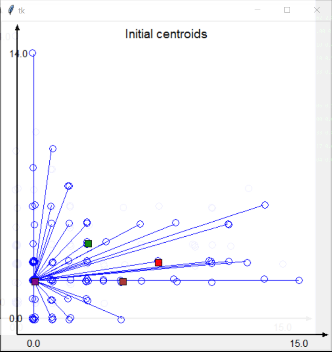
***Cluster Formation using Jaccard Similarity as a distance metric:***

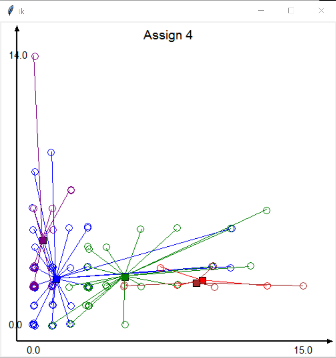
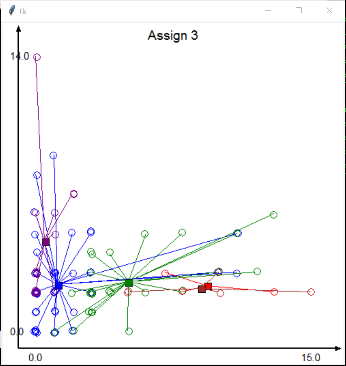
******

******

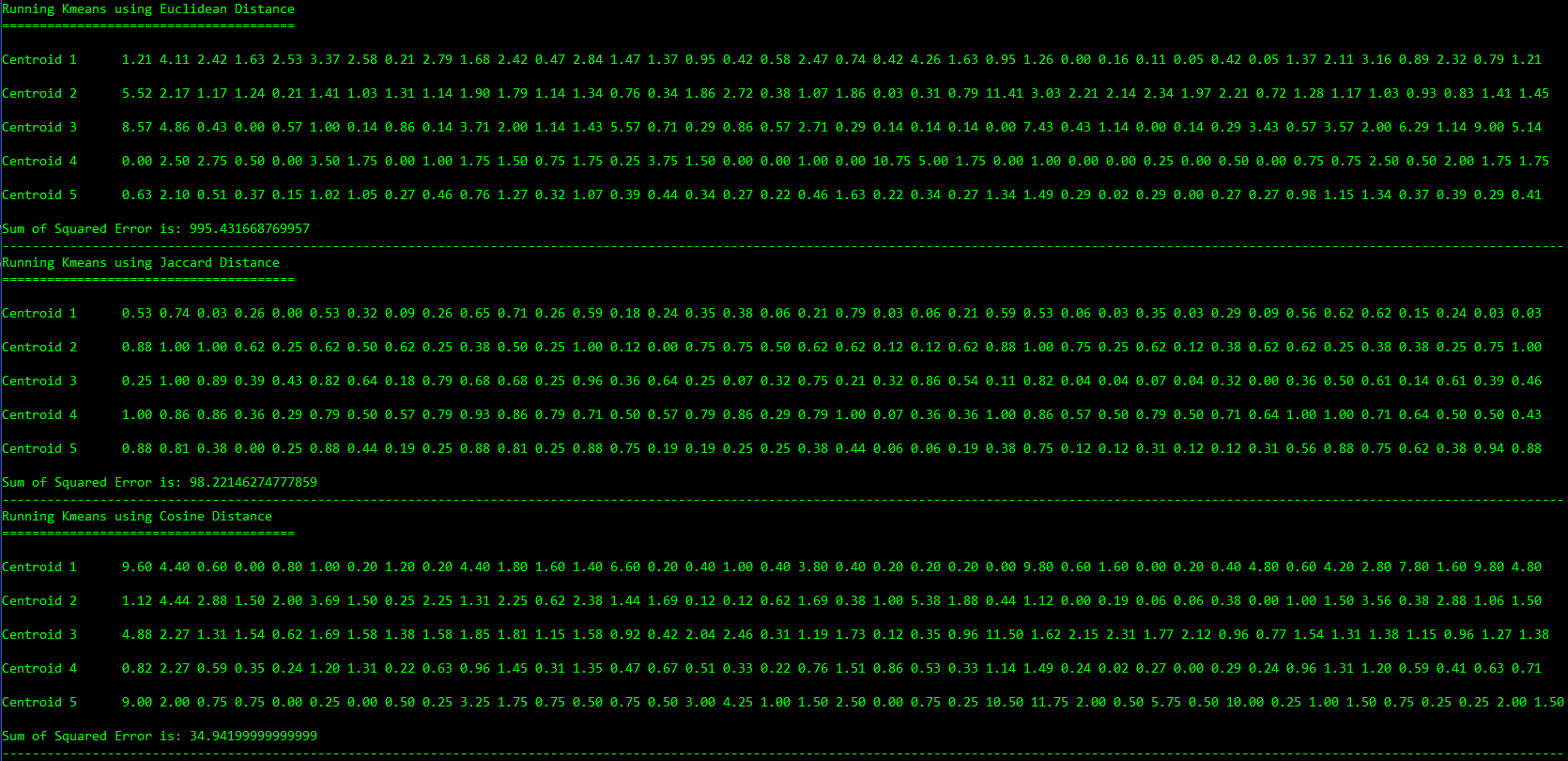
******

***Cluster Formation using Cosine Similarity as a distance metric:***

******

******

***Final Centroid Values of All the executions:***

******

1. **Interpretation & Analysis**

The data has been cleansed and proper features were extracted and their corresponding values were normalized to give accurate results. Cosine was the best due to low SSE value, formed clusters which predicted accurate results. Euclidean took many assignments to settle down to the final clusters. Jaccard had most different results mainly due to the input data which was an existence matrix rather than a frequency matrix like the earlier versions. All in all, K-means is one of the best clustering algorithm mainly because it’s very easy to understand with just a short algorithm.

***Comparison:***

The best way to compare the cluster formation is by calculating the SSE:

|  |  |  |
| --- | --- | --- |
| **Distance Metric** | **SSE** | **Prediction Rate (%)** |
| Cosine Distance | 34.4444 | 78 |
| Jaccard Distance | 97.714 | 56 |
| Euclidean Distance | 1027.976 | 45 |
| Classic K-Means Distance | 11076.2323 | 51 |

According to this we can deduce that Cosine has the low SSE suggesting that it predicts the clusters accurately. Below is the graph that visualizes the above table.

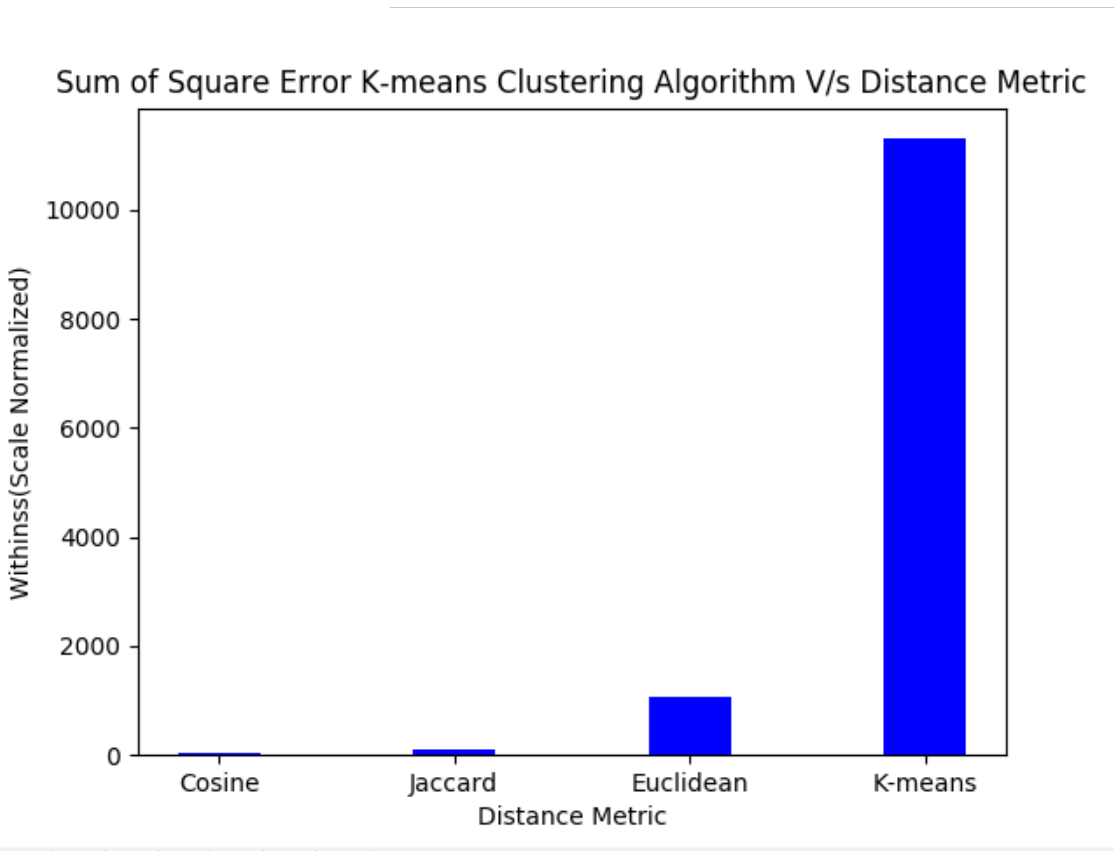


Figure: SSE Interpretation