**PROJECT – Evaluation of Clustering Algorithms**

Sergei Rogov

U231N0051

University of Nicosia

COMP-344 (Section 01): Machine Learning and Data Mining II

## Instructor: Dr Ioannis Katakis

January 5, 2024

Goals: getting acquainted with a variety of clustering algorithms, comprehending their outcomes, comparing the results and finding out which algorithm is the most suitable for a particular case.

Summary:

**Datasets**

**Dataset 1 – Countries**

This dataset contains information regarding 167 countries of the world. It has 9 attributes, which are: country - name;

child\_mort - death of children under 5 years of age per 1000 live births;

exports - exports of goods and services per capita. Given as %age of the GDP per capita;

health - total health spending per capita. Given as %age of GDP per capita;

imports - imports of goods and services per capita. Given as %age of the GDP per capita;

income - net income per person;

inflation - the measurement of the annual growth rate of the total GDP;

life\_expec - the average number of years a new born child would live if the current mortality patterns are to remain the same;

total\_fer - the number of children that would be born to each woman if the current age-fertility rates remain the same;

gdpp - the GDP per capita. Calculated as the total GDP divided by the total population.

In total, they are 2 integer, 1 string and 7 real attributes.

This dataset might be really interesting to analyze and to cluster because many people find it useful to know what is the best place to live and what is not, which countries are fully developed and which are less advanced.

[**https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data**](https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data)

**Dataset 2 – Songs**

The following dataset contains 57650 instances of songs of different artists and genres. The attributes are: artist (string); song name (string); link to a song (string) and, most importantly, lyrics (string). I am only interested in lyrics attribute because of the fact that I want to find some patterns in music lyrics and to see there any distinct clustering going on based on genre, artist, nationality of an artist, or simply on a time the songs were released. This task could be further transformed into another unsupervised learning activity, which is building a recommendation system.

[**https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/data**](https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/data)

**Dataset 3 – Customers**

The dataset includes details on the purchases made by 2,000 people from a specific area when they visited a real "FMCG" store. The loyalty cards people use to pay with have been used to capture all of the data. There are no missing values because the data has undergone preprocessing. The dataset has 8 attributes:

ID (integer) - shows a unique identificator of a customer;

Sex (0 - male, 1 - female);

Marital status: 0 - single, 1- non-single (divorced / separated / married / widowed);

Age (integer) – from 18 to 76;

Education: 0 – other, 1 – high school, 2 – university, 3 – graduate school.

Income (real): self-reported annual income in US dollars of the customer. From 35832 to 309364

Occupation: 0 - unemployed / unskilled, 1 - skilled employee / official, 2 - management / self-employed / highly qualified employee / officer.

Settlement size: 0 - small city, 1 – mid-sized city, 2 – big city.

This customer dataset, in my opinion, is really representative of what one might receive as a task on a job – company wants to customers want to

[**https://www.kaggle.com/datasets/dev0914sharma/customer-clustering**](https://www.kaggle.com/datasets/dev0914sharma/customer-clustering)

**Clustering Algorithms**

**K-Means**

K-Means is a clustering algorithm that initially requires a parameter k which represents the number of clusters.

K-Means algorithm is the following:

Step 1: k initial centroids are randomly chosen across the dataset.

Repeat:

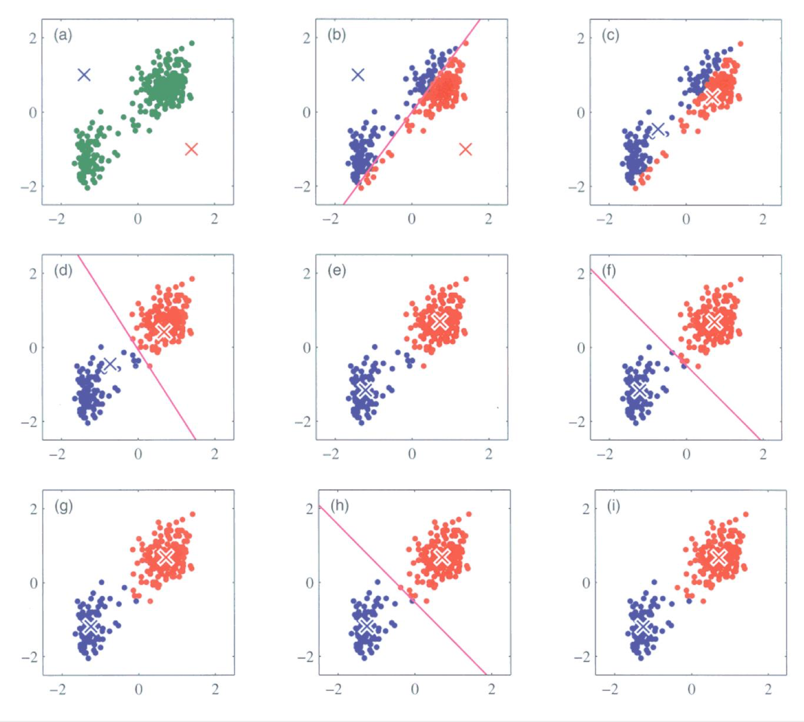
Step 2: each data instance is assigned a cluster by assigning it to the closest centroid.

Step 3: centroids are recomputed by finding the “center of mass” of each current cluster.

Until centroids do not change (or change insignificantly, for example too few instances change their assigned clusters).

Other stopping conditions might be used, like limiting the number of iterations. This could work well in many scenarios because almost all convergence happens during first few iterations. This method is also useful because of the fact that algorithm might stuck in an infinite loop.

Advantages of the K-Means are that it is easy to implement and easy to interpret results. It is computationally fast efficient as well. The disadvantage, however, is that the number of clusters is not known in the beginning, so it is not clear which value of parameter k to choose. Besides, as the initial centroids are chosen randomly, it leads to different clustering on the same dataset from one run of the algorithm to another. Moreover, K-Means does not work well when it comes to dataset containing non-spherical clusters.



**Hierarchical (Agglomerative)**

Hierarchical clustering is a clustering algorithm that is able to provide you with not only a set of clusters but with a whole folder-like structure of nested clusters, depicting the sequence of merges between data instances.

The algorithm of agglomerative clustering is described here:

Step 1: proximity matrix is computed

Step 2: each data instance is considered to be a separate cluster

Repeat:

Step 3: merge two closest clusters

Step 4: recompute the proximity matrix

Until all clusters are merged into one single cluster.

There several approaches available for updating the proximity matrix, them being: taking the minimum distance between clusters, the maximum distance, the average distance, the distance between centroids and many others.

The problem with hierarchical clustering is that no objective function is being directly minimized. Besides, if a decision to split or merge clusters is made, this action cannot be undone. The advantage is that it is not needed to specify the number of clusters to run the algorithm. It is possible to get any desired number of clusters by cutting the dendrogram (tree representation of the clustering) at a proper level.

**DBSCAN**

Density-based Spatial Clustering of Applications with Noise is a clustering algorithm that is capable of finding clusters based on its points density as well as finding and eliminating outliers

DBSCAN algorithm:

Step 1: find neighbors of every point within the radius epsilon and identify core points (data points with over the specified number of points in the radius around it).

Step 2: find connected components of core points, not considering any non-core ones.

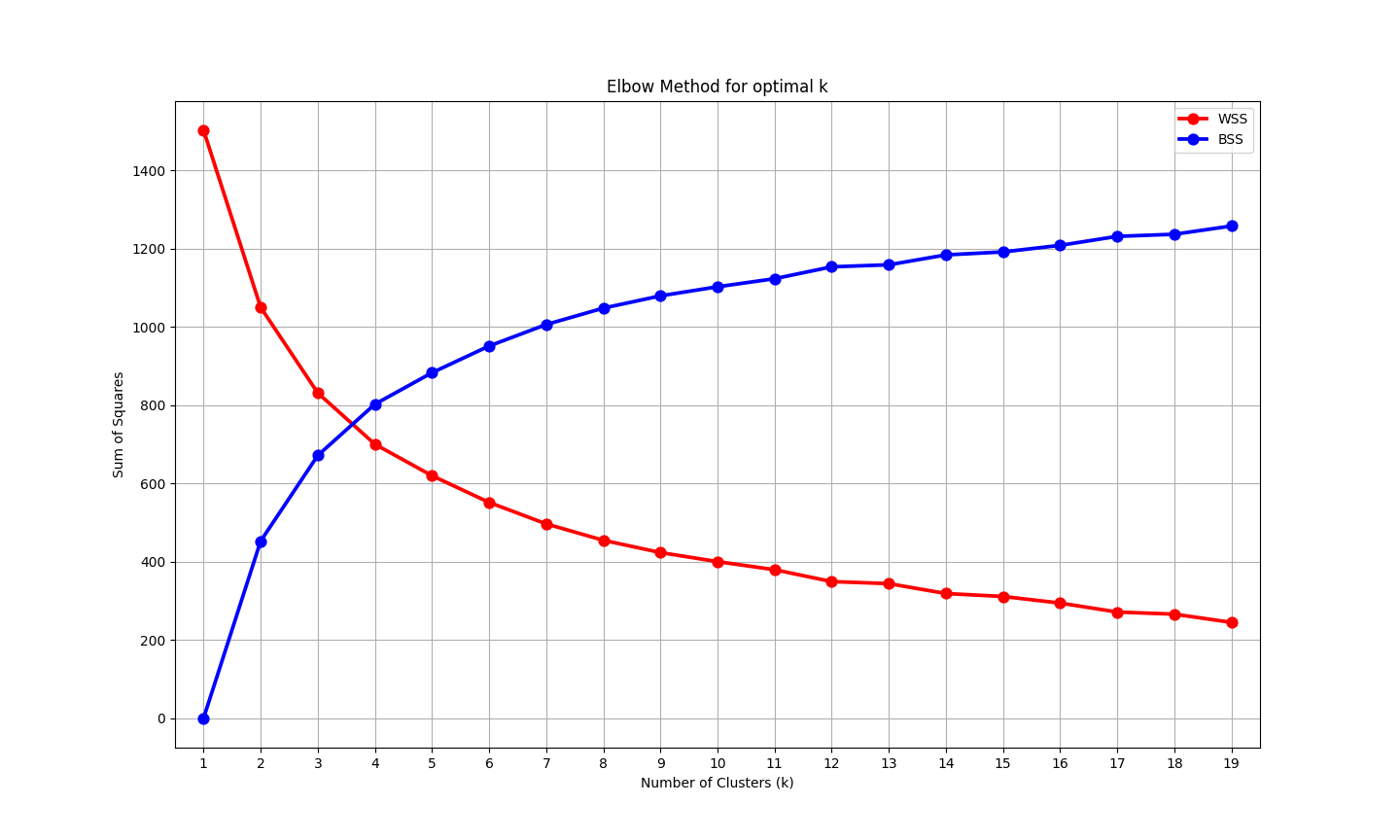
Step 3: assign each non-core point to a nearby cluster if the core point from this cluster is in the epsilon radius, otherwise assign it to noise.

DBSCAN works really well with complex cluster shapes and it is good with dealing this outlier. DBSCAN, however, does not work well with varying densities and high-dimensional data.

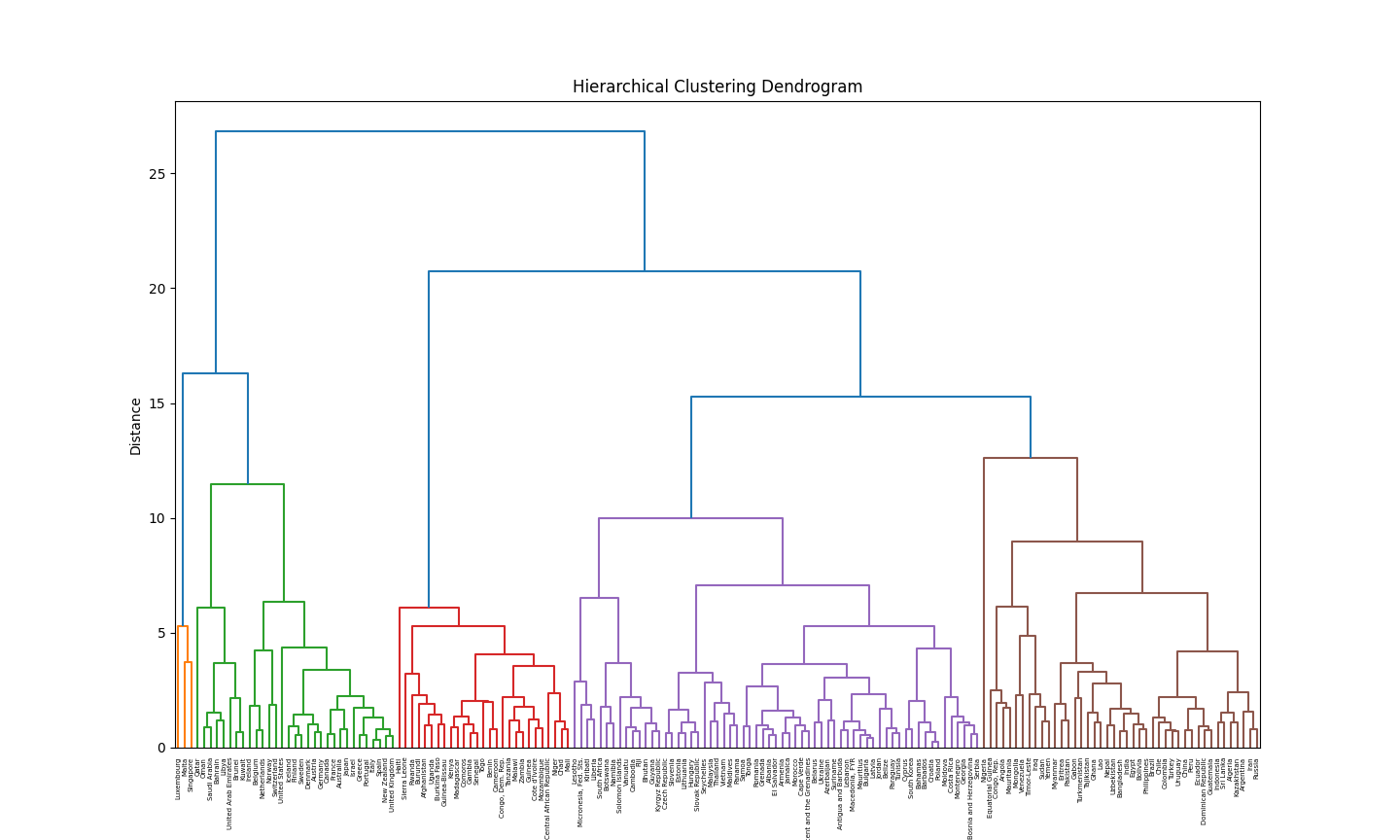
**Results**

**Countries dataset**

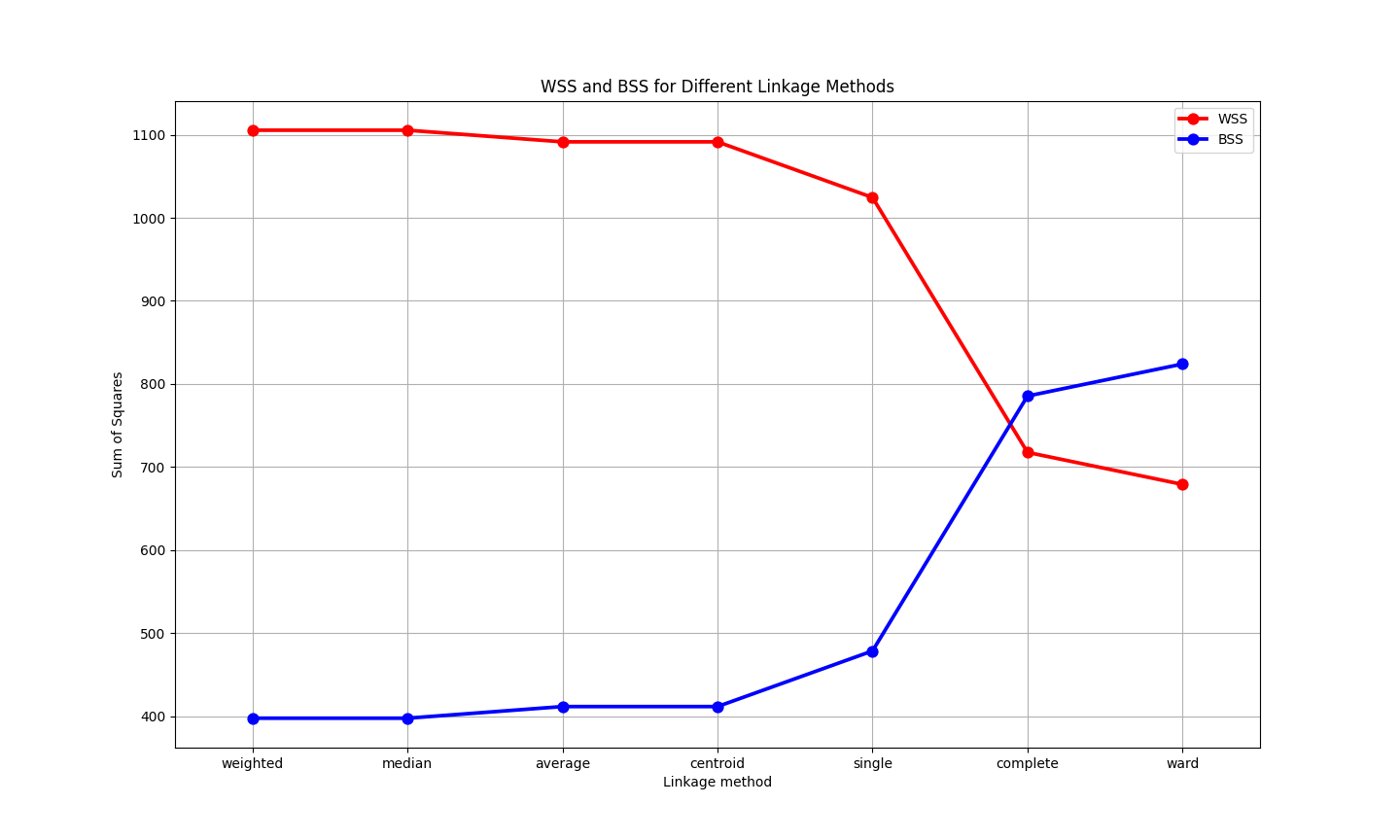
K-Means, Sum of Square errors with different number of clusters.

****

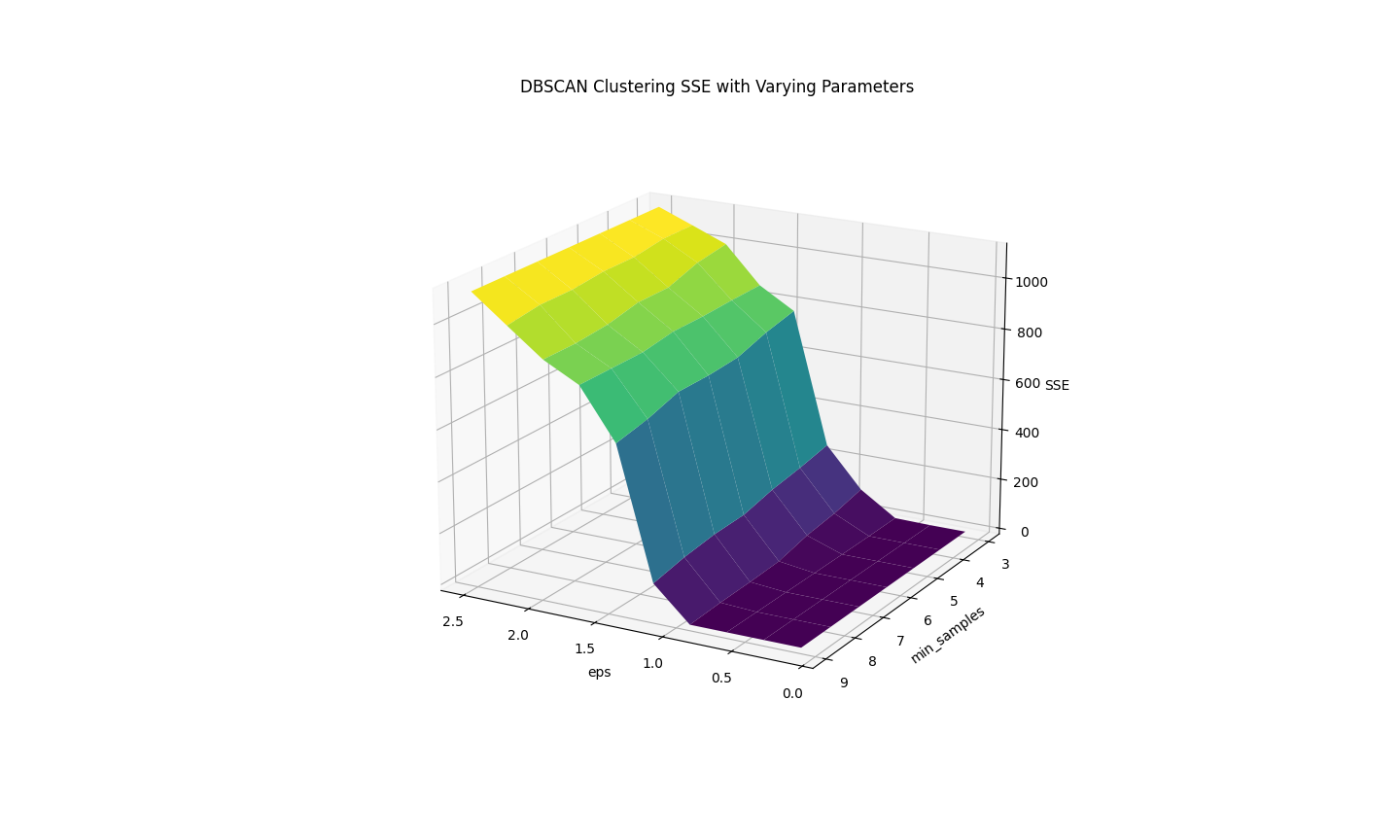
Dendrogram, Hierarchical Clustering.

****

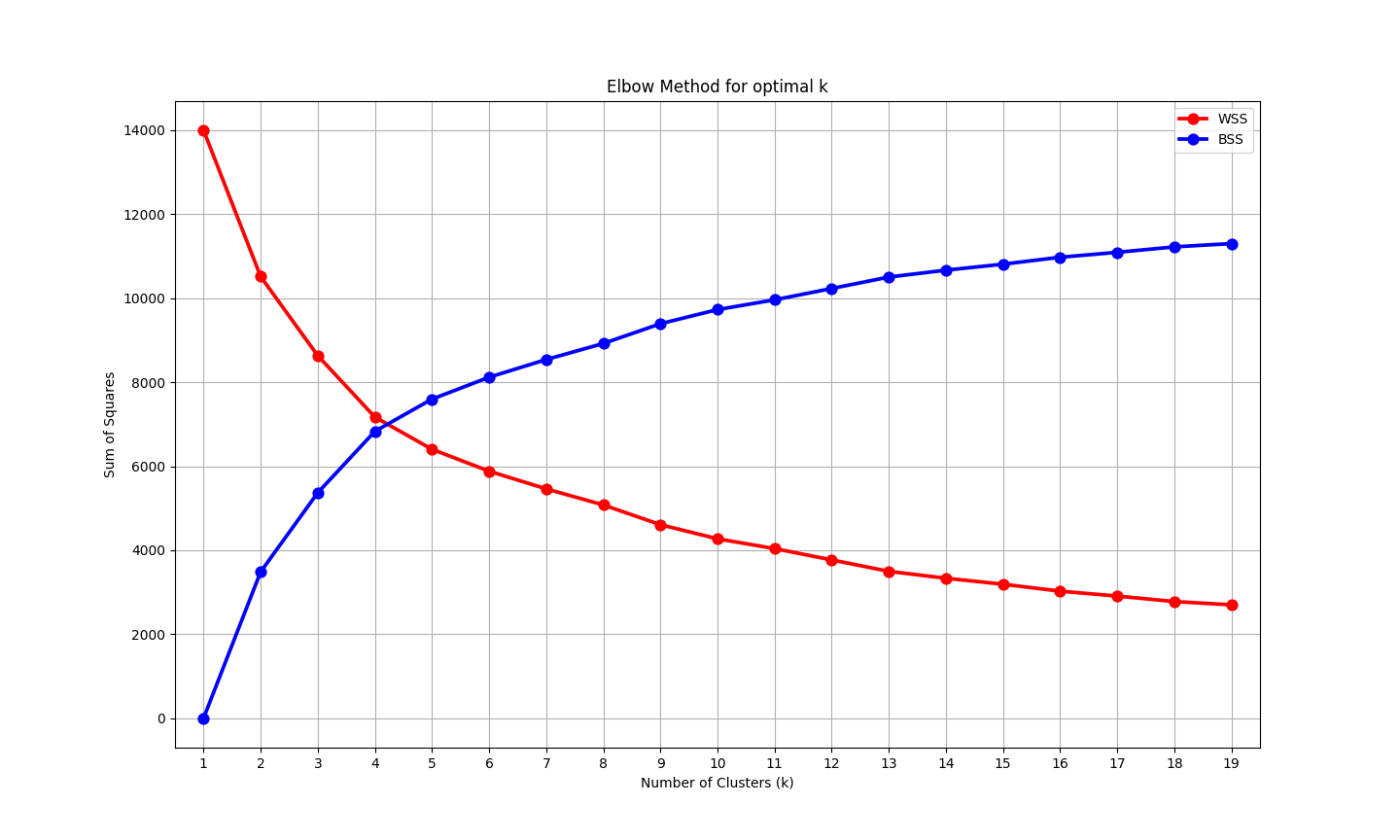
Hierarchical Clustering, Sum of Square errors with different linkage methods. The level on where to cut the dendrogram was inspected visually.

****

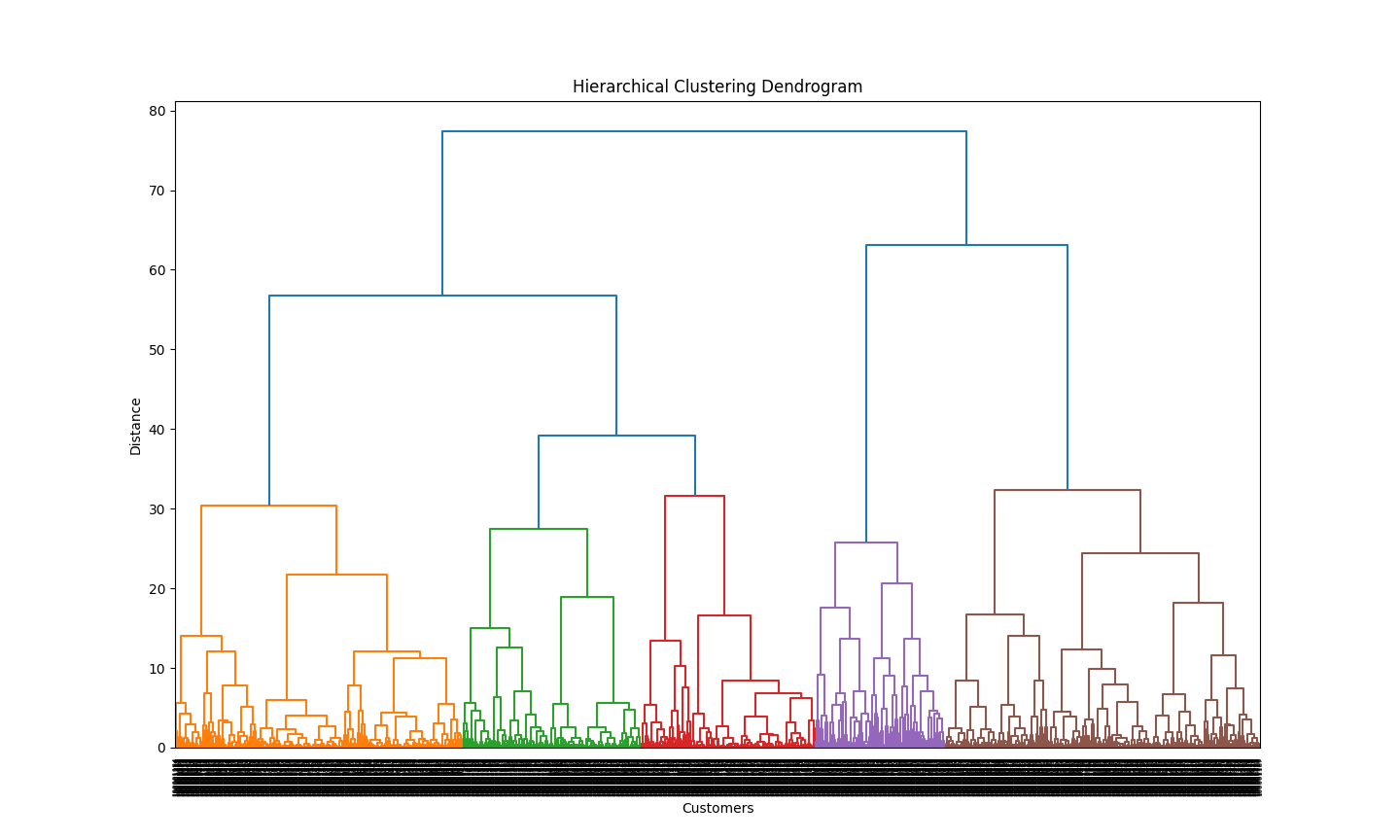
DBSCAN, Sum of Square errors with different epsilon and min\_points

****

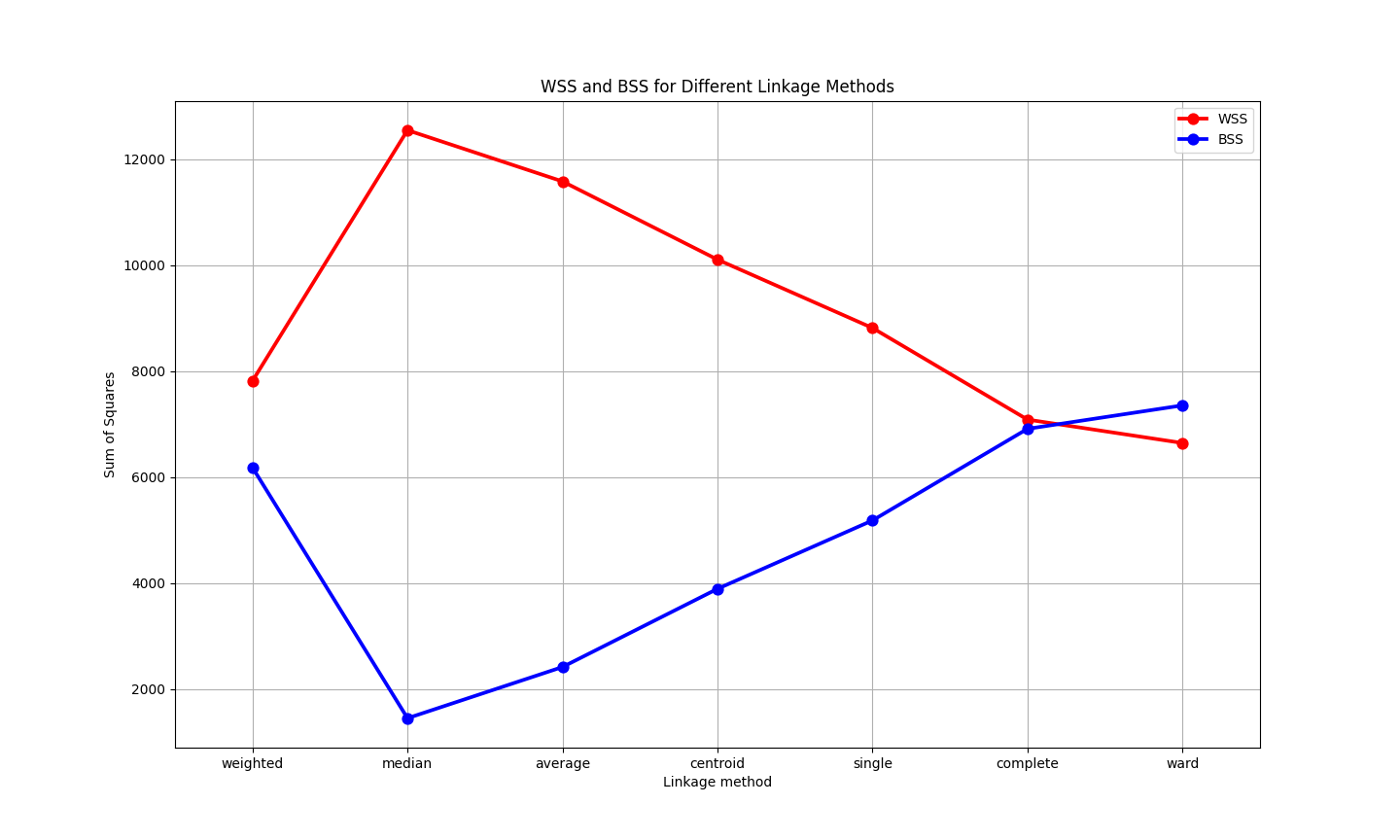
**Customers dataset**

K-Means, Sum of Square errors with different number of clusters.

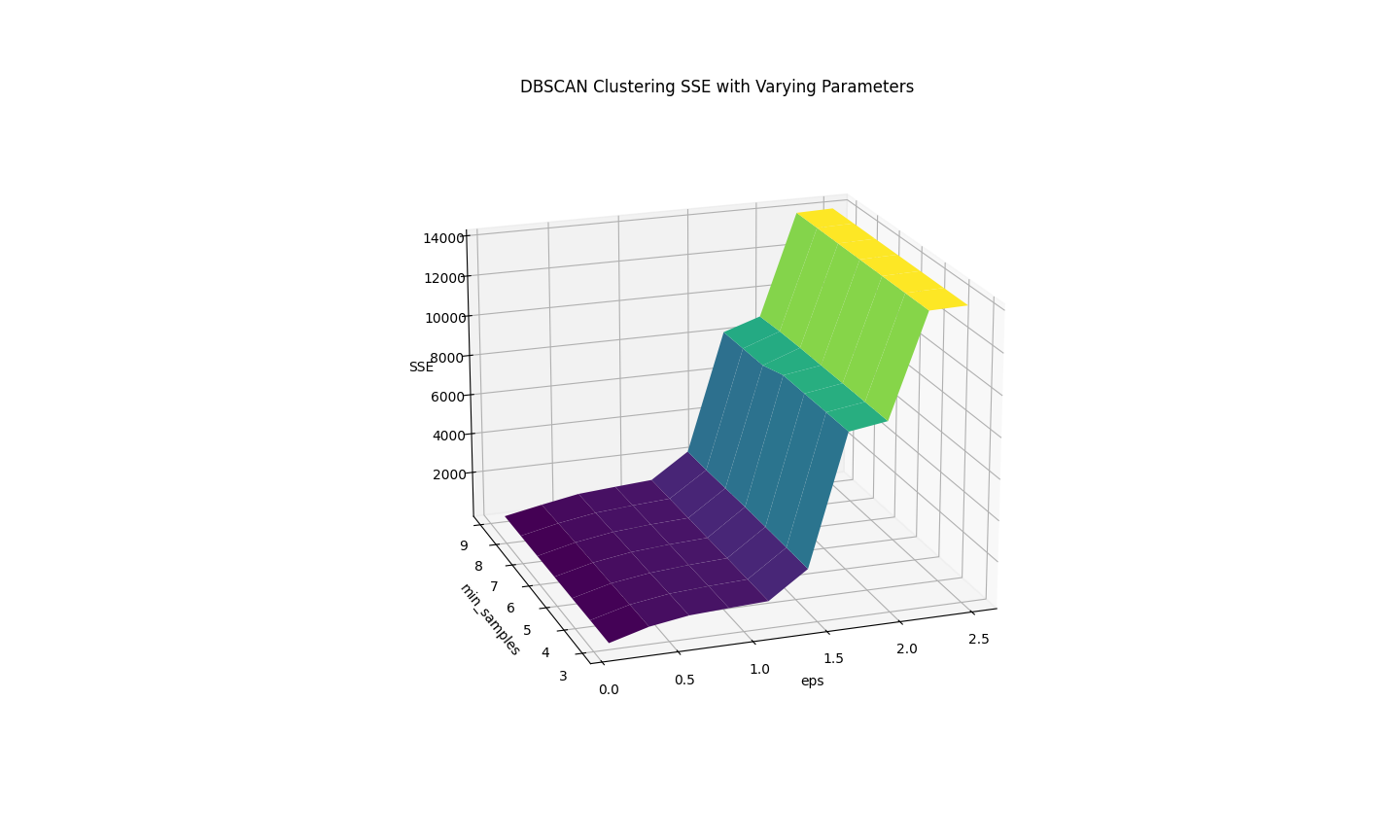
Dendrogram, Hierarchical Clustering.

****

Hierarchical Clustering, Sum of Square errors with different linkage methods. The level on where to cut the dendrogram was inspected visually.

****

DBSCAN, Sum of Square errors with different epsilon and min\_points

****

**Discussion**

**Conclusions**

**References**

[**https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data**](https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data)

[**https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/data**](https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/data)

[**https://www.kaggle.com/datasets/dev0914sharma/customer-clustering**](https://www.kaggle.com/datasets/dev0914sharma/customer-clustering)