CZ4032 Data Analytics and Mining

Project 1: A Study on Methods of Clustering using K means, K means++ and DB Scan

Group 26

Name	Matric No.
SIM XIN NI CLODIA	U2021577B
LIM YI JIE JASMIN	U2021622K
LIM ZHENGWEI TREVOR	U2022401K
SANKAR SAMIKSHA	U2021021D
TEO HAN HUA	U2022463B

Table of Contents

ABSTRACT

Clustering methods span far and wide – Centroid-based, density-based, distribution-based and hierarchical. Selecting the most appropriate method is not entirely straightforward. Comparisons between each method must be considered in regard to dataset and objective. To help with this, this report documents our approach and execution of various clustering methods: K-means, K-means++ and DB SCAN. Usage of optimal parameters are explored: Elbow Curve and Silhouette Scores are used for the optimal number of K clusters in K-Means and K-Means++, and K-NearestNeighbours is used for Epsilon in DB Scan as well. We also explore other parameters and aids such as normalisation. We evaluate the strengths and weaknesses of each method and their efficacy. We illustrate these with an experiment set out in this paper, using a database of houses in Miami.

1 INTRODUCTION

1.1 MOTIVATION FOR CLUSTERING

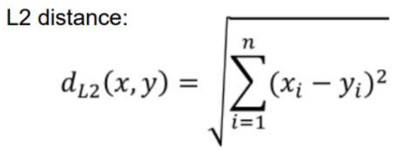
Astronomical amounts of data are collected daily. This is considered raw data and needs to be mined, processed, and analysed before it turns into useful data, capable of aiding decision making. Clustering is a part of the processing phase - it is an unsupervised machine learning task which is used to categorise data into patterns or groups. It helps to analyse patterns and relationships within a large dataset [1]. Based on different objectives, varying methods can be employed. However, each method has its own set of pros and cons. In this paper we mainly examine a centroid-based method, K-means, and a density-based method, Density-Based Spatial Clustering of Applications with Noise (DB Scan). We will compare the strengths and weaknesses of the methods, and how varying parameters will affect the result.

1.2 OBJECTIVE

The objective of our report is to analyse the strengths and weaknesses of the different clustering methods. We will compare the methods' performances based on the selected dataset and on factors such as the speed of the method and how well the method clusters.

1.3 SCOPE

For our experiment, we will only be covering three methods, namely K-means, K-means++, and DB Scan. Since there are many distance metrics used for clustering, we have narrowed our experiment to using only the Euclidean Distance.



L2 distance:

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Figure 1: Euclidean Distance (From Lecture Notes)

2 RELATED WORK

Numerous papers are written to describe and analyse existing clustering methods. In this section, we review some related papers to understand each method, before performing our own comparisons on our selected methods (K-Means, K-Means++, and DB Scan).

2.1 K-MEANS

K-means is one of the simplest and most popular unsupervised clustering methods. It takes an unlabelled dataset and groups them into different clusters, based on the similarity of the samples. K-means is a centroid-based algorithm, meaning each cluster generated is associated with a centroid. The initial centroids in K-means are chosen at random. Using a distance metric - the algorithm will categorise the data points into different clusters by minimising the sum of squared distances in each cluster.

In K-means, constant K refers to the number of centroids needed in the dataset. This must be set manually, and the elbow curve is usually employed to find the most optimal K. Then, a random initialisation of centroids is set according to K, and the algorithm starts. Using the Euclidean Distance, in our case, data points are assigned to a centroid closest to them. In every iteration, a new cluster centroid is generated by computing the mean of each cluster. The iteration goes on until there are no changes of cluster assignments compared to the previous iteration. When this happens, we say that the algorithm has converged, and the best clustering case is generated.

2.2 K-MEANS++

One major drawback of K-means is that it is very reliant on the initialization of the clusters[2]. As such, K-means++ improves the cluster selection such that the initial clusters are as far away from each other as possible[2], [3]. The steps to conduct the initialisation of clusters are as follows[2], [3]:

```
Select a random data point as the first cluster
do {
for each $x$ data point in the dataset $D$, calculate the minimum squared distance between $x$
and the current clusters defined thus far $G(x)$.
Select the $x$ data point with the highest $G(x)$ as the new centroid
} while (we do not have $k$ number of clusters)
```

2.3 DB SCAN

In contrast to the centroid-based K-means and its sub variants, DB Scan is density-based. Density-based clustering prevents us from being restricted to a set constant K value (number of clusters). Instead, it creates regions corresponding to a cluster of data points that fall within a set density level. Regions that are below that level will be merged into a single cluster. This threshold level is a combination of epsilon (distance threshold) and MinPts (density threshold) in DB Scan[4]. This allows for data points that are not reachable by core points of any cluster to be categorised as a "noise point", thus preventing the clusters from being skewed by outliers.

3 METHODS

We analyse the clustering methods by using a dataset of housing information from Miami. We process the data using Python and visualise it using Jupyter Notebooks. All source codes are available at: https://github.com/S-Samiksha/ClusteringAnalysis.

3.1 DATASET

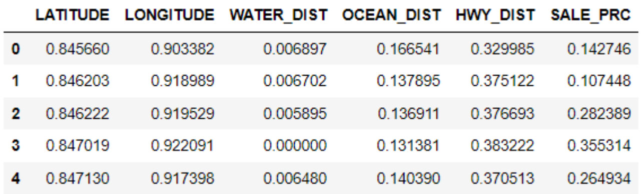
The publicly available dataset is taken from https://www.kaggle.com/code/juniorbueno/miami-florida-prediction-prices/data?select=miami-housing.csv. It contains data of single-family homes sold in Miami. A few variables were narrowed to show the difference in results obtained between K-means, K-means++ and DB Scan. There is no actual purpose of predicting data into clusters. The objective of this project is merely to show how DB Scan is different in clustering than K-means and K-means++.

Additionally, further exploration of K-means and K-means++ in 3-dimensional data was done simply to see how K-means and K-means++ differs in clustering. For simplicity, the optimal K for the 3-dimensional data was set to 4.

The following variables were chosen in the project. The definition of the variables are as follows (taken from the Kaggle website):

HWY_DIST	Distance to the nearest highway (an indicator of ambient sound noise) (feet)
WATER_DIST	Distance to the nearest body of water (feet)
OCEAN_DIST	Distance to the ocean (feet)
SALE_PRC	Sale price ($)
LATITUDE	Co-ordinate specifying north-south position of a point
LONGITUDE	Co-ordinate specifying east-west position of a point

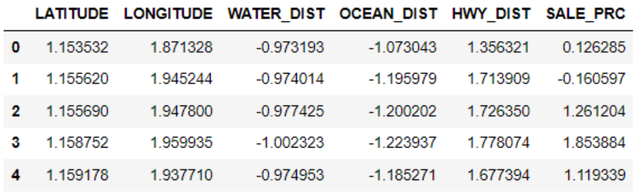
3.1.1 Effects of scaling on the dataset



	LATITUDE	LONGITUDE	WATER_DIST	OCEAN_DIST	HWY_DIST	SALE_PRC
0	0.845660	0.903382	0.006897	0.166541	0.329985	0.142746
1	0.846203	0.918989	0.006702	0.137895	0.375122	0.107448
2	0.846222	0.919529	0.005895	0.136911	0.376693	0.282389
3	0.847019	0.922091	0.000000	0.131381	0.383222	0.355314
4	0.847130	0.917398	0.006480	0.140390	0.370513	0.264934

Figure 2: MinMax Scaler for the data

MinMax Scalar resets the feature values such that they are in the range 0 to 1.



	LATITUDE	LONGITUDE	WATER_DIST	OCEAN_DIST	HWY_DIST	SALE_PRC
0	1.153532	1.871328	-0.973193	-1.073043	1.356321	0.126285
1	1.155620	1.945244	-0.974014	-1.195979	1.713909	-0.160597
2	1.155690	1.947800	-0.977425	-1.200202	1.726350	1.261204
3	1.158752	1.959935	-1.002323	-1.223937	1.778074	1.853884
4	1.159178	1.937710	-0.974953	-1.185271	1.677394	1.119339

Figure 3: Standard Scaler for the data

Standard Scaler resets the feature values such that the mean is 0 and scales the values to the unit variance. The standard value of the data point, x, is calculated as[9], [10]:

$$z = \frac{x - mean\ of\ samples}{standard\ deviation\ of\ samples}$$

3.2 K MEANS AND K MEANS++

3.2.1 Deciding optimal K

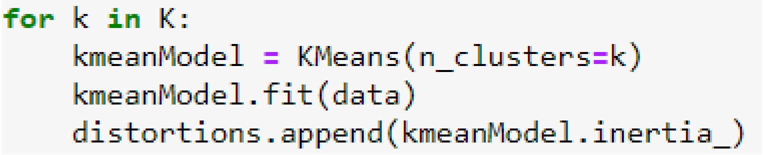
When looking at a set of data, the number of clusters can be observed visually. However, frequently, it is necessary to have a mathematical approach to find the number of clusters because it is not always possible get an accurate number of clusters by visualization only[5]. This is especially so when there are more than 3 variables used to cluster data which cannot be represented visually in a graph.

Hence, an elbow curve is used to help determine the number of clusters that would give the best clustering for K-means. There are two values that are important to draw the elbow curve – distortion and inertia. Distortion is defined as “the average of the squared distances from the cluster centres of the respective clusters”[5]. Euclidean Distance is usually used to calculate distortion. Inertia is defined as “sum of squared distances of samples to their closest cluster centre”[5].

For the purposes of this project, the Sci-Kit Learn is used to provide the inertia calculation. The formula used for inertia is:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \left(\left\|x_i - \mu_j\right\|\right)^2$$

Where the μ_j is the mean of the samples in cluster and x is the data points in the cluster. This formula is described as the within-cluster sum-of-squares criterion[6].



```
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(data)
    distortions.append(kmeanModel.inertia_)
```

Figure 4: Partial Code for Plotting Elbow Curve

Figure 4 shows the code for plotting the elbow curve. For every number of clusters from 1 to 10, the inertia is obtained. It is then plotted into an Elbow Curve. However, at times, the elbow of the curve is difficult to visualize. Hence, the following method is adopted[7]:

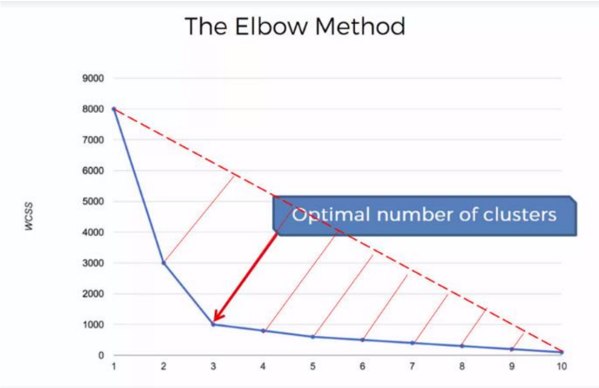
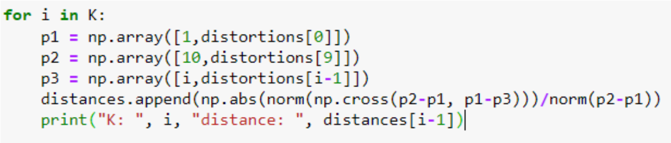


Figure 5: Finding Elbow

In the above Figure, an imaginary line is drawn from the first point to the end point. The longest perpendicular distance to this imaginary line implies that is the optimal number of clusters to use. The below Figure shows the code used:

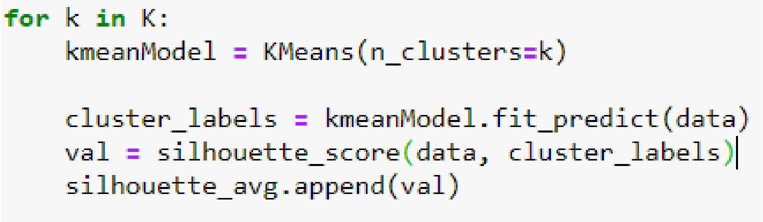


```
for i in K:
    p1 = np.array([1,distortions[0]])
    p2 = np.array([10,distortions[9]])
    p3 = np.array([i,distortions[i-1]])
    distances.append(np.abs(norm(np.cross(p2-p1, p1-p3)))/norm(p2-p1))
    print("K: ", i, "distance: ", distances[i-1])
```

Figure 6: Finding the Distance between Imaginary Line and Point

The distances are found using the Numpy Library in python. The number of clusters that yields the largest perpendicular distance to the imaginary line, is the optimal number of clusters.

Although the above method provides a good mathematical way to determine the optimal number of clusters, a much better way would be to use the Silhouette Score provided by Sci-Kit Learn. According to Sci-Kit Learn, the Silhouette Score indicates "how close each point in one cluster is to the points in the neighbouring clusters"[8]. From a given range of clusters, K, explored, the number of clusters, k, with the highest Silhouette score indicates the optimal number of clusters k. Below shows the code used:



```
for k in K:
    kmeanModel = KMeans(n_clusters=k)

    cluster_labels = kmeanModel.fit_predict(data)
    val = silhouette_score(data, cluster_labels)
    silhouette_avg.append(val)
```

Figure 7: Silhouette Score Code

3.3 DB SCAN

As mentioned in section 3.3, DB Scan eliminates the need for setting the number the clusters as in K-means and K-means++. Instead, we have to decide the Epsilon and MinPts.

3.3.1 Optimal Epsilon

Calculating the optimal Epsilon is integral - if the value is too large, a cluster will take into account more points than it should, resulting in larger clusters. On the flip side, if Epsilon is too small, it will cause some clusters to be split apart into more clusters[11]. As a result, epsilon must be varied according to the user's objectives.

To calculate the optimal number for Epsilon, we use K-NearestNeighbour distances (average distance of every point to its K-Nearest Neighbours), then sort the distance in ascending order to obtain the distance of the K-th Nearest Neighbour. The K is estimated to be the square root of the number of datapoints. We then plot the K-distance graph[12]. From the k-distance graph, the "elbow" point in the graph is the threshold whereby anything above can be deemed as too far and hence not part of the cluster. The corresponding distance is our Epsilon.

3.3.2 Optimal MinPts

In determining MinPts, we choose 4. Experiments show that, where K is larger than 4, the corresponding K-Distance graph does not differ significantly from a 4-Distance one[4]. Later on, however, we notice other factors affecting this decision in the Experiments section.

4 Experiments and Results

For these experiments, we have decided to compare only 5 variables from the dataset that we have found, namely LATITUDE, LONGITUDE, TOT_LVG_AREA, OCEAN_DIST, and HWY_DIST.

4.1 K-means and K-means++ Results

In accordance with the two methods set out in section 3.2.1, the optimal number of K is usually 4 for any given pair of variables stated. To give an example, this section explores LATITUDE and LONGITUDE.

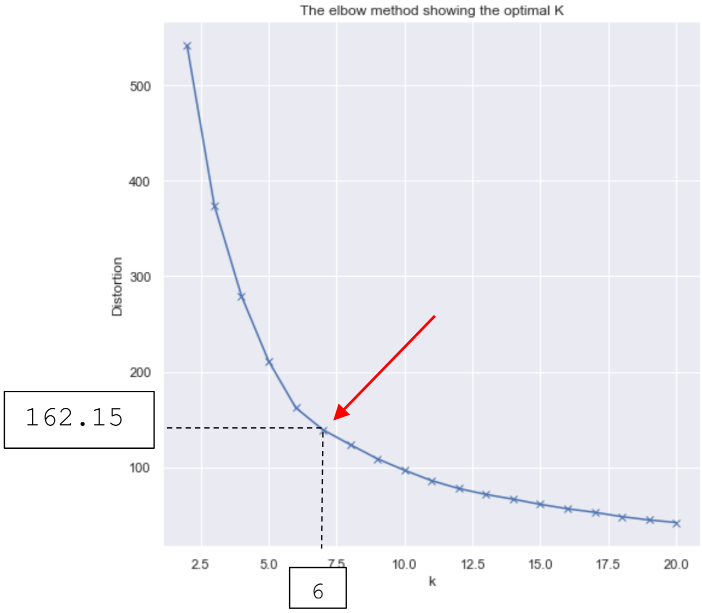
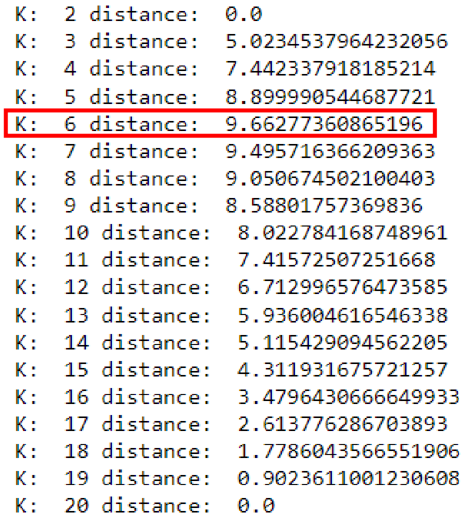


Figure 8: Elbow curve for variables LONGITUDE and LATITUDE

In the above figure, it is difficult to tell exactly where the elbow in the curve is. Using the imaginary line method, the optimal number of clusters obtained is 6. The below figure shows the distances obtained:



```
K:  2 distance:  0.0
K:  3 distance:  5.0234537964232056
K:  4 distance:  7.442337918185214
K:  5 distance:  8.899990544687721
K:  6 distance:  9.66277360865196
K:  7 distance:  9.495716366209363
K:  8 distance:  9.050674502100403
K:  9 distance:  8.58801757369836
K:  10 distance:  8.022784168748961
K:  11 distance:  7.41572507251668
K:  12 distance:  6.712996576473585
K:  13 distance:  5.936004616546338
K:  14 distance:  5.115429094562205
K:  15 distance:  4.311931675721257
K:  16 distance:  3.4796430666649933
K:  17 distance:  2.613776286703893
K:  18 distance:  1.7786043566551906
K:  19 distance:  0.9023611001230608
K:  20 distance:  0.0
```

Figure 9: Distances Calculated from Imaginary Line for LONGITUDE vs LATITUDE

Using the Silhouette Score, we obtained the below graph:

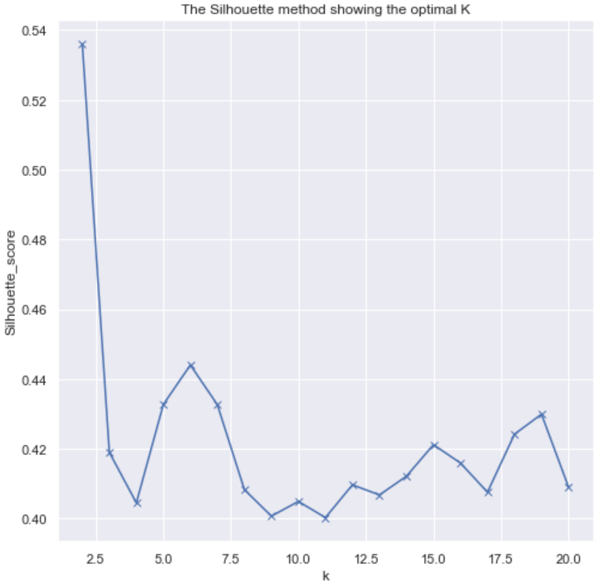


Figure 10: Silhouette Score for LONGITUDE vs LATITUDE

The graph shows a peak at k = 6. Hence, the optimal number of clusters is 6. In this scenario, the two methods yielded the same results. However, this was not always the case especially when the data has no proper spherical clusters.

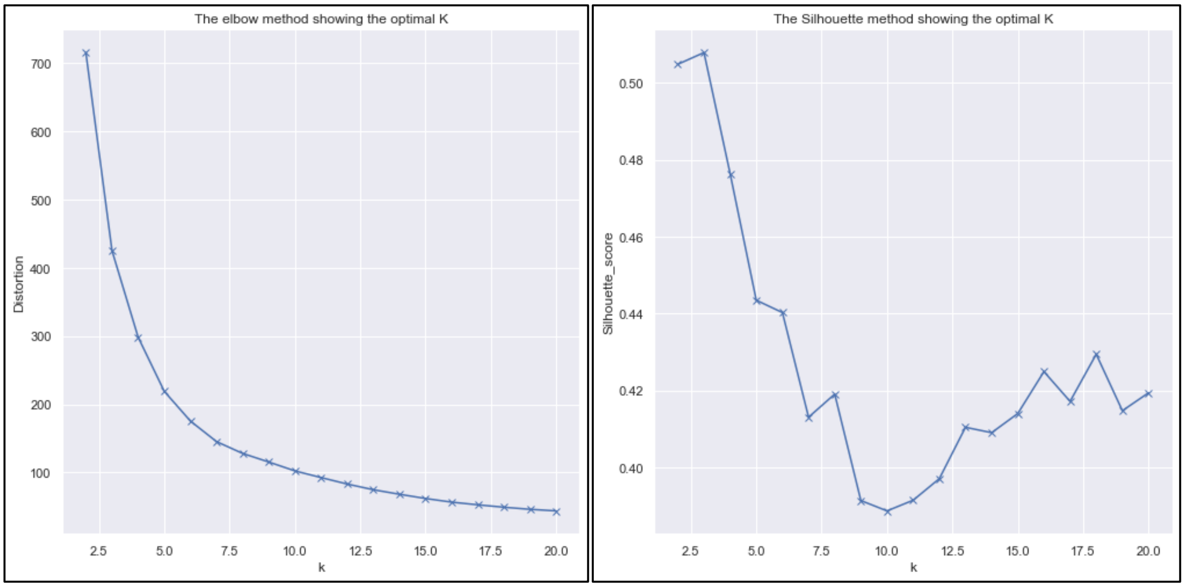
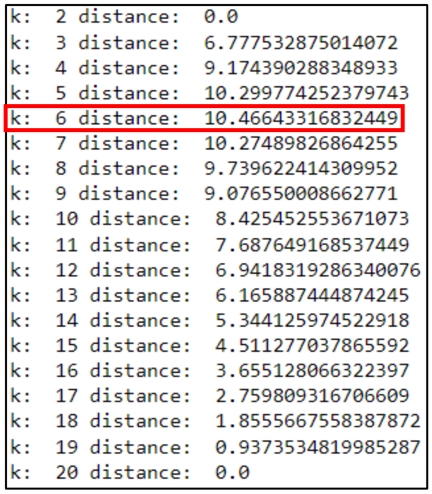


Figure 11: Elbow curve and Silhoutte for OCEAN_DIST vs WATER_DIST



```
k:  2 distance:  0.0
k:  3 distance:  6.777532875014072
k:  4 distance:  9.174390288348933
k:  5 distance:  10.299774252379743
k:  6 distance:  10.46643316832449
k:  7 distance:  10.27489826864255
k:  8 distance:  9.739622414309952
k:  9 distance:  9.076550008662771
k:  10 distance:  8.425452553671073
k:  11 distance:  7.687649168537449
k:  12 distance:  6.9418319286340076
k:  13 distance:  6.165887444874245
k:  14 distance:  5.344125974522918
k:  15 distance:  4.511277037865592
k:  16 distance:  3.655128066322397
k:  17 distance:  2.759809316706609
k:  18 distance:  1.8555667558387872
k:  19 distance:  0.9373534819985287
k:  20 distance:  0.0
```

Figure 12: Distances from Imaginary Line for OCEAN_DIST vs WATER_DIST

For the variables OCEAN_DIST vs WATER_DIST, the imaginary line method suggests k = 6 but the Silhouette method suggest k=8. The elbow curve itself is very unclear if one were to do it visually. Hence, k=6 to k=8 were explored.

The results of the rest of variables are available in the ipynb submitted together with this report.

4.2 DB Scan Results

4.2.1 Optimal Epsilon Using K-NearestNeighbours

In accordance with the method set out in 3.3.1, K is the square root of the number of rows we have in the dataset. With 13932 records in the database, K should be roughly 118. Below is our k-distance graph

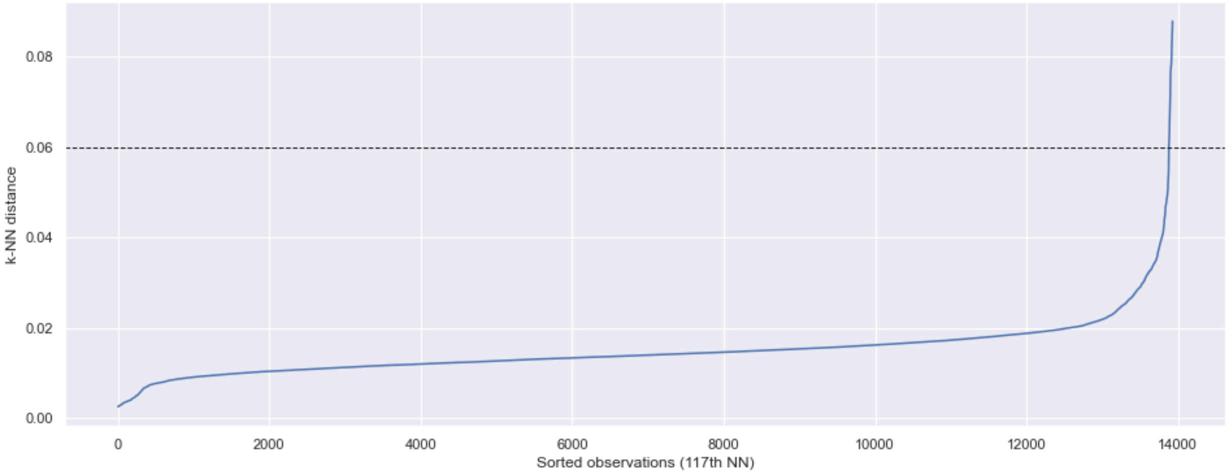


Figure 13: k-NN Distance Graph to Determine Epsilon

4.2.2 Performing DB Scan

We take Epsilon to be 0.06 as determined in 4.2.1, and minPts as 4 as determined in 3.3.2.

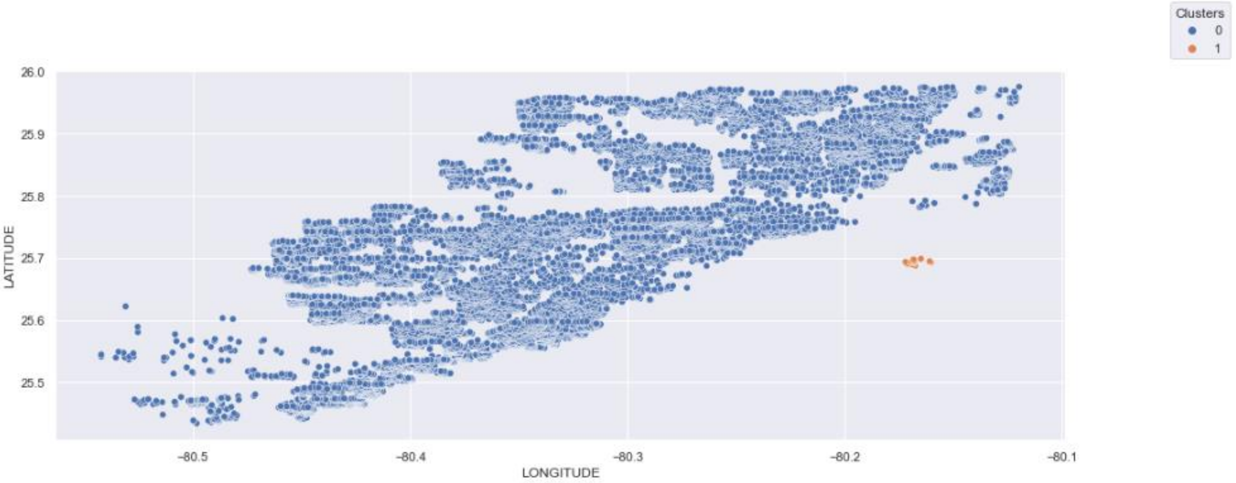


Figure 14: "Optimal" DB Scan

As seen above, this is the result of an "optimal" DB Scan according to existing papers. Here we realise that "optimal" is not ideal. We have to consider the objective of this clustering. If the purpose is to find households that may be affected by an outbreak of a disease, then having only 2 clusters and no noise points tells us the localisation and containment of such. This clustering would thus be useful. However, if the purpose is to find how many hospitals should be constructed to maximise accessibility, then this clustering would not have taught us anything. For the sake of objectives similar to the latter, we proceed to explore more scans using "non-optimal" parameters.

4.2.3 Normalising Data

We reperform 4.2.1 and 4.2.2 on normalised data, using both standard (Z-score) and min-max normalisation. Although our respective Epsilon values differ, we find that we ultimately achieve the same results.

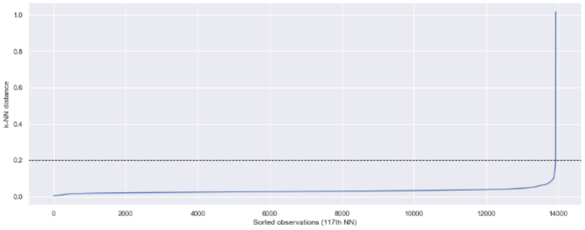


Figure 15: Min-Max Normalised k-NN

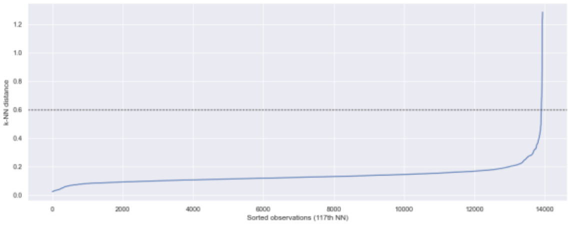


Figure 16: Standard Normalised k-NN

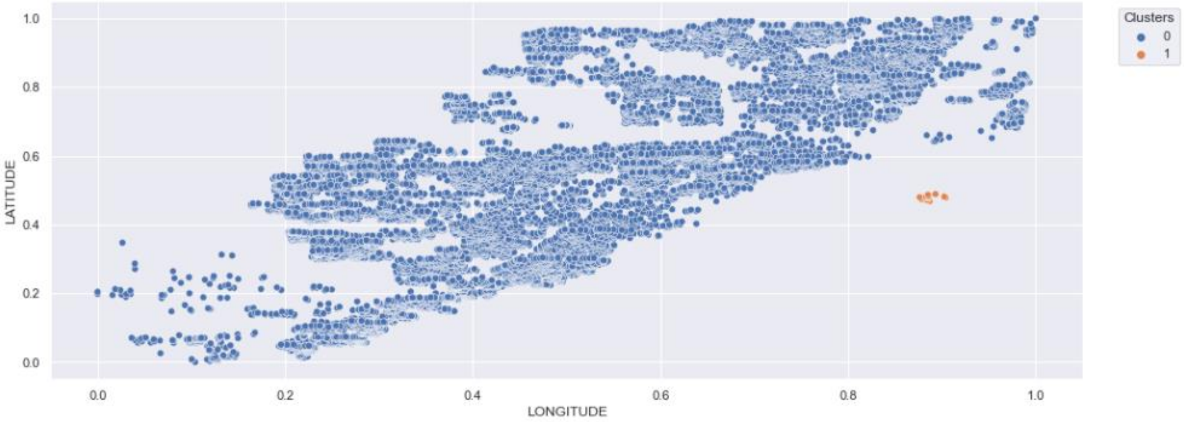


Figure 17: Min-Max Normalised DB Scan

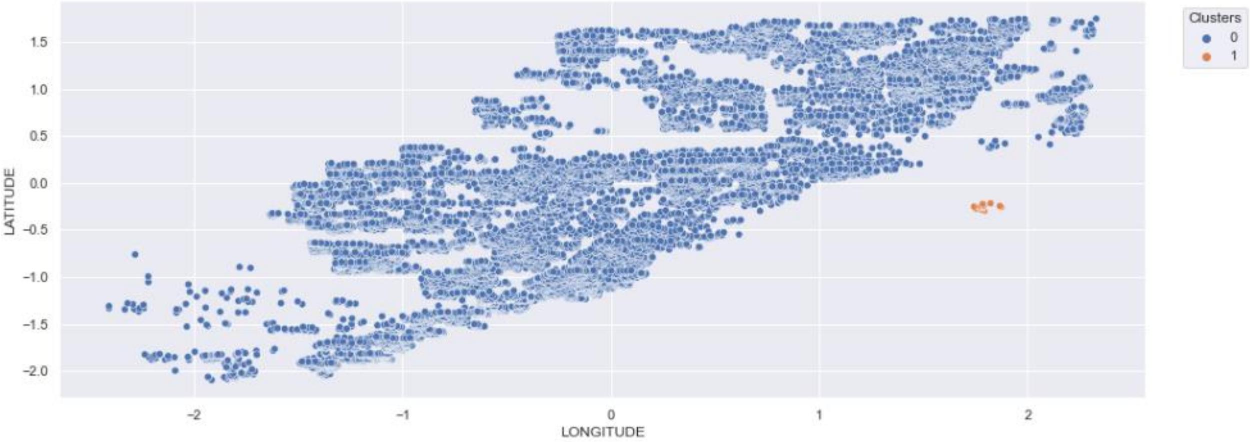


Figure 18: Standard Normalised DB Scan

This is not surprising on closer inspection. We take our Epsilon value from the normalised k-NN graph – Epsilon is hence also scaled by a factor. When using it on the respective DB Scans, the factor is similar to its normalisation and thus the results should be the same, as seen above.

4.2.4 Varying Epsilon Parameter

In line with our 2nd purpose mentioned in 4.2.2, we first attempt to find a more ideal Epsilon value.

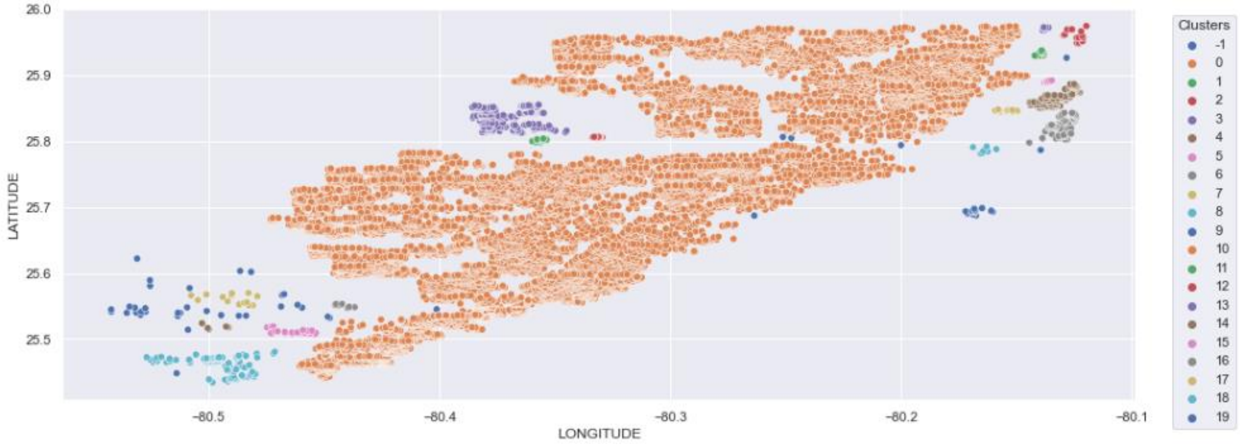


Figure 19: DB Scan with Epsilon = 0.0085, minPts = 4

Here we can see the impact that reducing Epsilon has. Since the distance to a core point required to be considered part of a cluster has reduced, more points are no longer grouped together. In essence, clusters have now been split. Note that this has also given rise to noise points, as they are not close enough to be grouped with any cluster. However, we now notice that there are many clusters. We try to reduce that in our next experiment.

4.2.5 Varying MinPts Parameter

To reduce the number of clusters, we should increase the minimum number of points required in order to form a cluster.

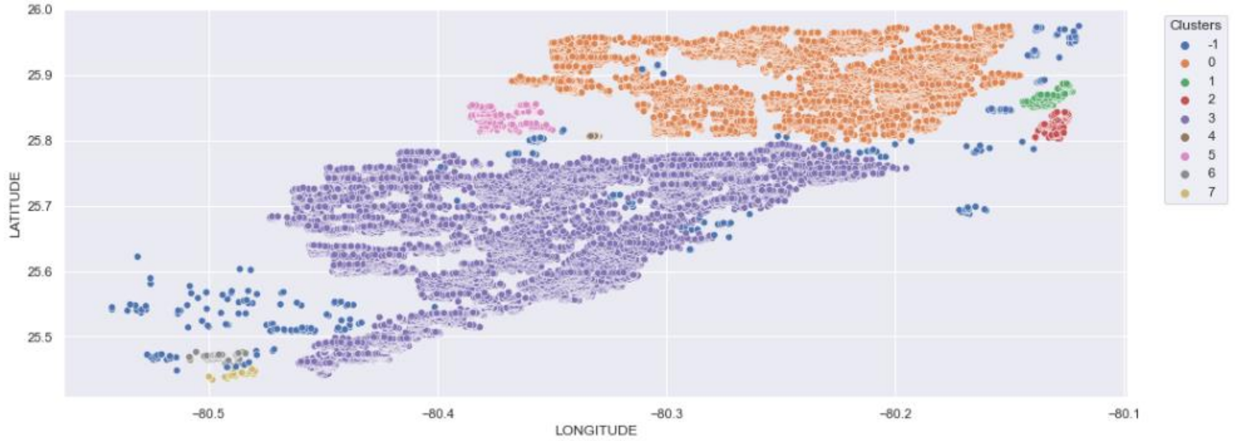


Figure 20: DB Scan with Epsilon = 0.0085, minPts = 20

In this scan, we arrive at 8 clusters, which seem a more likely solution for our purpose of finding how many hospitals to build.

4.3 NORMALISATION OF DATA

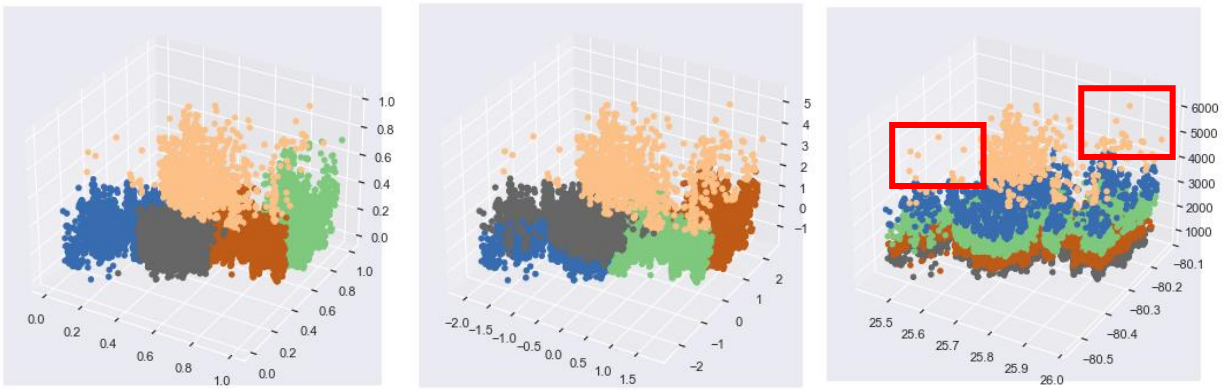


Figure 21: K-means using Longitude, Latitude, TOT_LVG_AREA using MinMax Scaler, Standard Scaler, no Scaler

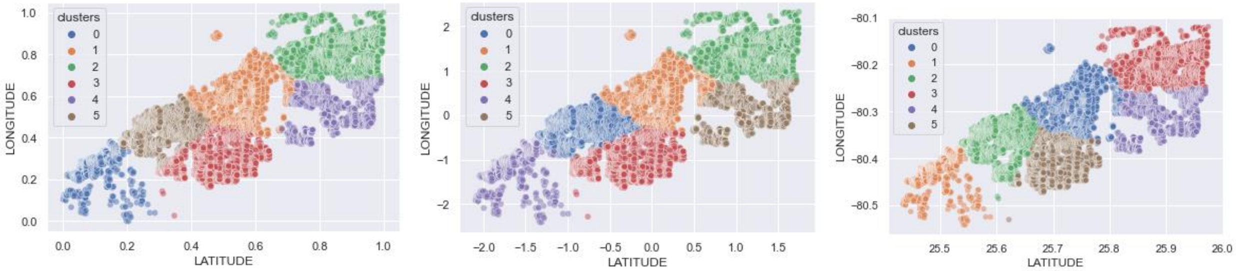


Figure 22: K-means using Longitude, Latitude using MinMax Scaler, Standard Scaler, no Scaler

The scaling does not affect 2-dimensional data for K-means significantly. However, for 3-dimensional data, the results are very different. When there is no scaling applied in the 3-dimensional K-means, the data is affected more by the outliers – these are highlighted by the red boxes. However, when scaled, the outliers are spread more evenly in the clusters.

We chose to use MinMax Scaler as we want to preserve the shape of the dataset. In this case, Standard Scaler would not be suitable as our data does not have a normal distribution, so using Standard Scaler would distort our dataset and make our K-means analysis inaccurate[13].

5 COMPARISON OF METHODS

5.1 K-MEANS VS K-MEANS++: HOW INITIALIZATION OF CENTROIDS AFFECT CLUSTERING

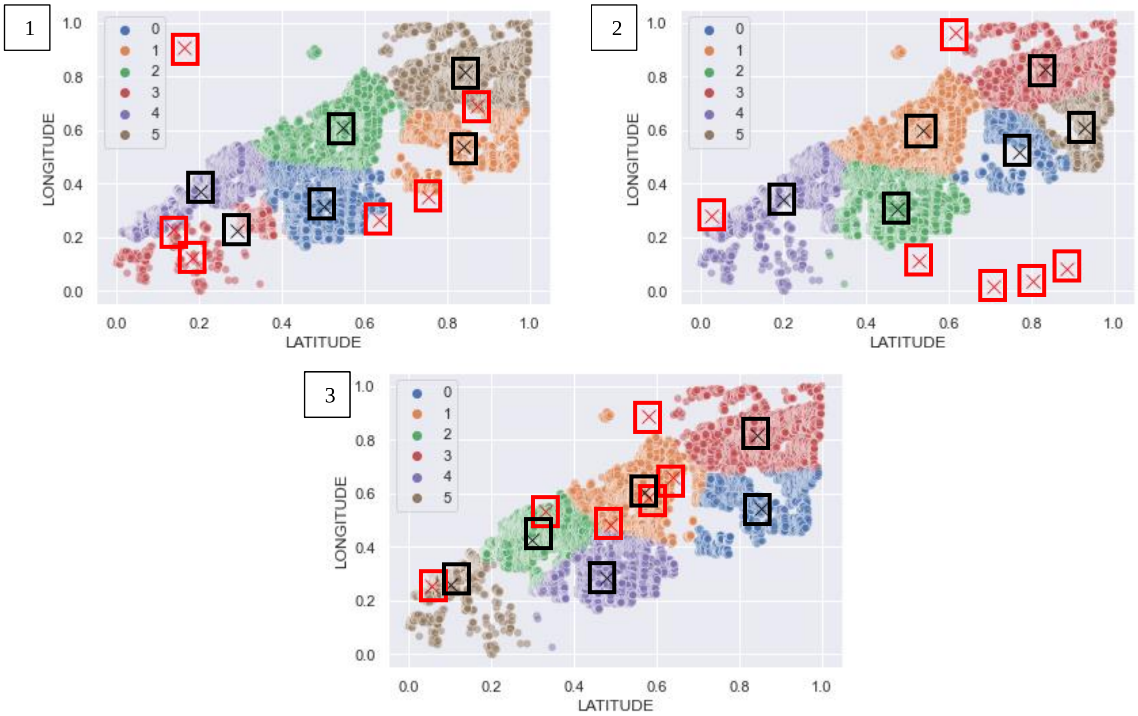


Figure 23: Initialization of Clusters for different runs of the code for Longitude and Latitude for K-means

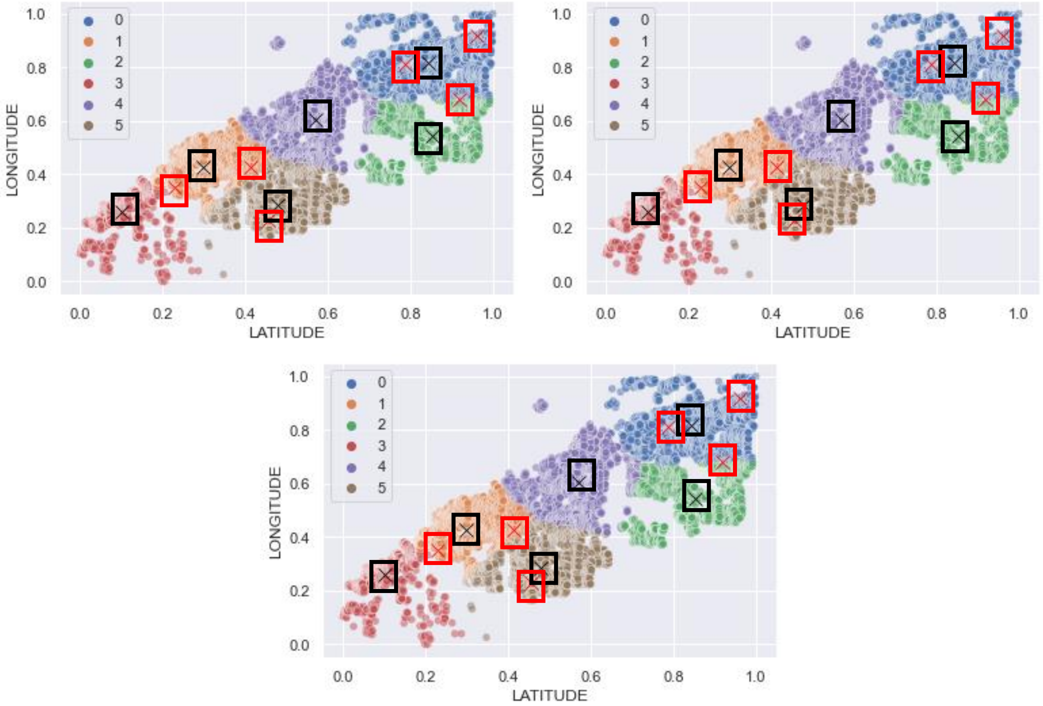


Figure 24: Initialization of Clusters for different runs of the code for Longitude and Latitude for K-means++

The red boxes and black boxes in Figure 21and Figure 22represent the initial and final centroids respectively. It is observed that initial centroids for K-means always changes every run and are not near any of the final centroids. However, for K-means++, initial centroids remain the same for each run and are near the final centroids. Additionally, the final centroids and clusters for K-means are different every run. This suggests that the initialization of centroids is very important in K-means.

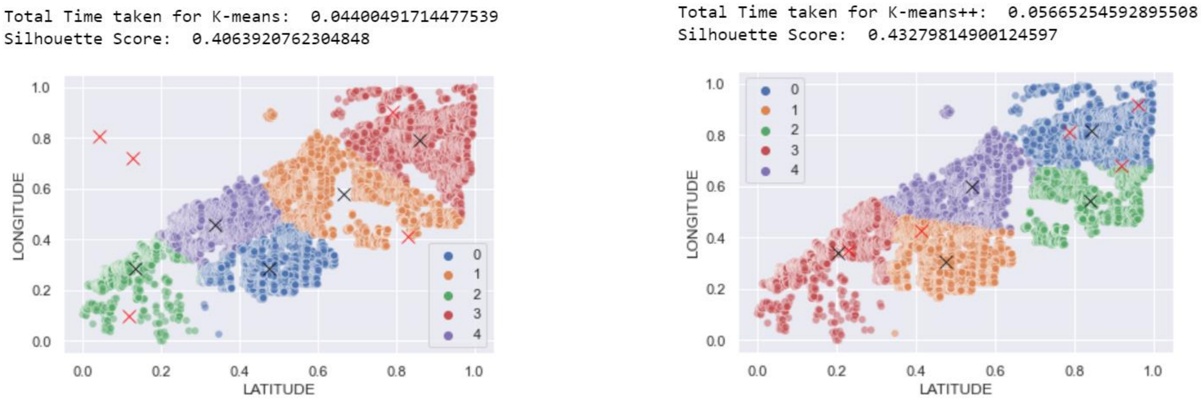
Run	Silhouette Score
1	0.39199740100664654
2	0.4047263620242516
3	0.44662785747117789

The silhouette score for K-means++ is 0.4468921705556768 for all runs. The worst value for Silhouette score is -1 while the best is 1. K-means obtains values that are both higher and lower than the K-means++. Hence, to obtain a consistent and relatively good result, K-means++ is preferred – we initialise the clusters using the formula described in section 2.2. In fact, K-means++ was able to reach the final clustering shown after 29 iterations while K-means took more than 90 (this value will change depending on the initialisation). These values were found by limiting the max_iter parameter in the fit_predict function. The source code and the results are available in the ipynb submitted together with this report. This analysis was only done for Longitude vs Latitude as it was sufficient to show the significance of the initialisation of centroids.

5.2 K-MEANS VS K-MEANS++: OVERHEAD OF INITIALISATION

During our experiments, we realised while K-means++ generally has better initialisation of centroids which leads to better clusters, as explained in 5.1, it may cause an overhead.

For example, when we initialise our K value to be 5, the time taken to run K-means++ is longer than that of K-means.



```
Total Time taken for K-means:  0.04400491714477539
Silhouette Score:  0.4063920762304848
```

```
Total Time taken for K-means++:  0.05665254592895508
Silhouette Score:  0.43279814900124597
```

Figure 25: Time taken for k-means and k-means++ when K value = 5

A possible reason for this is because of K-means++'s inherent sequential nature. [14] While K-means initialises centroids at random, K-means++ employs an algorithm which allows them to initialise centroids. This algorithm would require K-means to run through the whole data K times, and this would add extra time needed to run the clustering. As the dataset grows, the number of clusters naturally increases, and the number of iterations needed to determine the centroids increases, leading to K-means++ having a longer runtime as compared to K-means.

The complexity of the K-means algorithm is as follows[15]:

$$Complexity = \ O(tkn)$$

$$where\ t = number\ of\ iterations, k = number\ of\ clusters, n = number\ of\ data\ points$$

Whereas the complexity of the K-means++'s algorithm is as follows[15]:

$$Complexity = O(tk^2n)$$

$$where\ t = number\ of\ iterations, k = number\ of\ clusters, n = number\ of\ data\ points$$

However, despite the higher time complexity, K-means++ is generally more effective and gives much more accurate clusters, making up for its runtime. In rare cases where speed performance is prioritised over accuracy, K-means would be a better option.

5.3 K-MEANS AND K-MEANS++: LIMITATIONS IN CLUSTERING NON-SPHERICAL DATA

Both K-means and K-means++ would not work well on data with non-spherical shape[16]. K-means presume that the data is spherical and has a well-defined structure. When the dataset has many outliers, it will generate centroids that are far from legitimate data points, leading to clusters that are not accurate as the data points that should belong to the same cluster would be in different clusters[17]. Naturally, we would want to filter out these anomalies so that they do not affect the results. However, since K-means presumes that the data is structured, they would force these outliers to be part of a cluster. Many data points would end up being unrelated to the clusters that they are clustered into, and one or more centroids may be associated with a group that is similar.

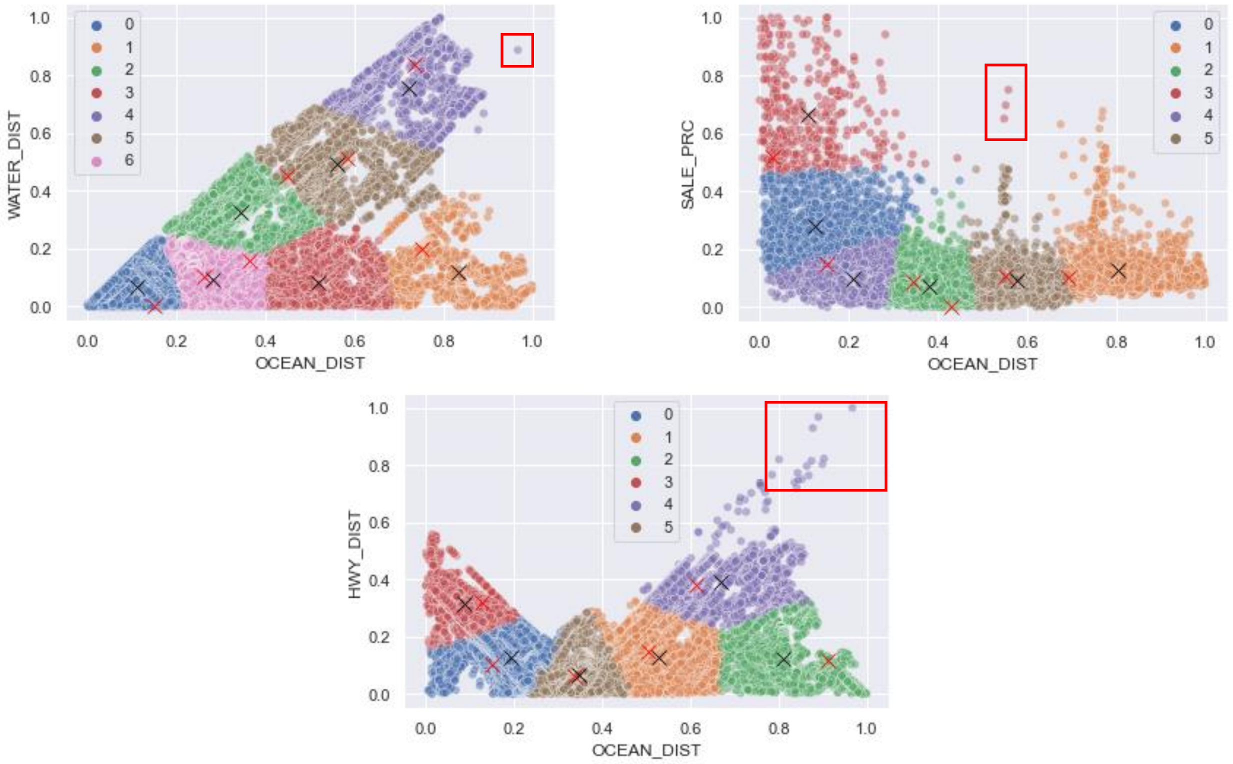


Figure 26: Outliers being forced into clusters in k-means++ for non-spherical data

5.4 K-MEANS AND K-MEANS++ VS DB SCAN

On the other hand, DB Scan handles outliers much better based on the nature of its algorithm. As a density-based method, DB Scan clusters the points based on the proximity of that point to an existing cluster. With the application of the parameters epsilon distance and minimum number of data points, anomalies would naturally be considered a non-neighbour of any clusters and not be categorised into any clusters. Hence, DB Scan would filter out anomalies and the clusters would be more accurate. We compare against Figure 26, using DB Scan.

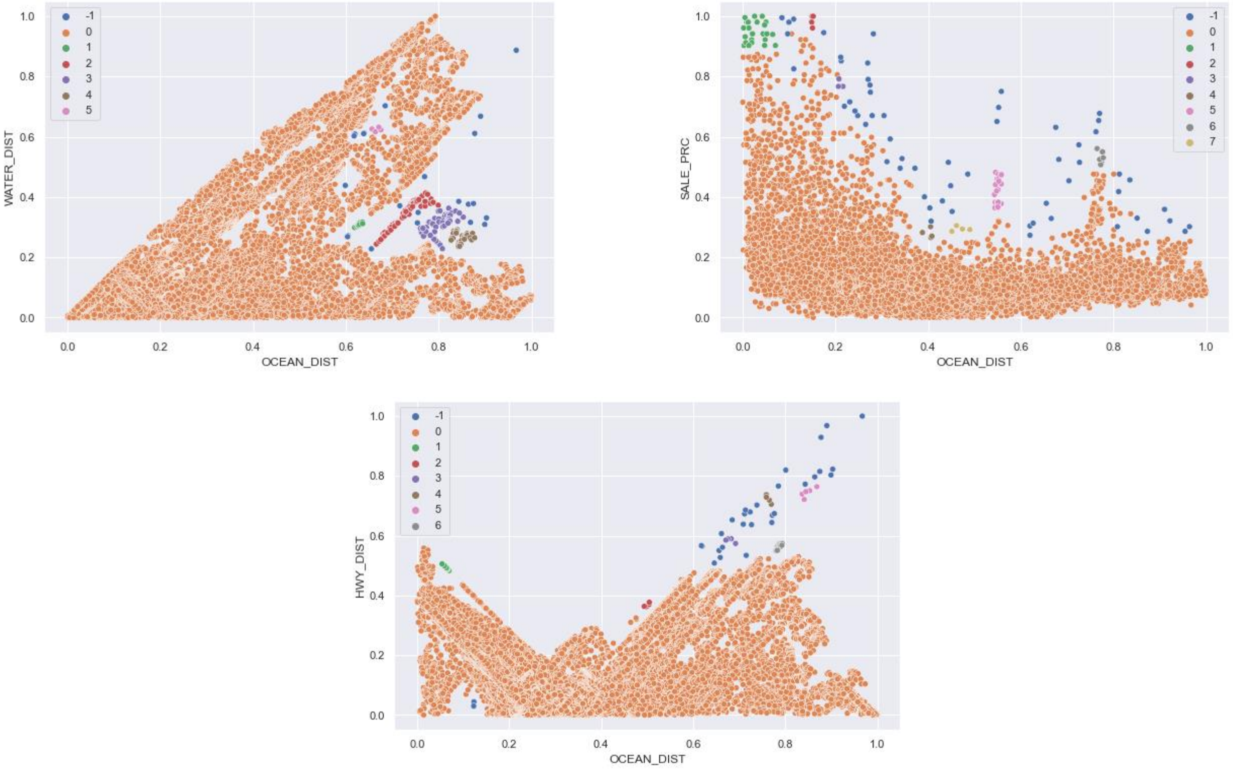


Figure 27: Outliers classified as noise points in DB Scan

We can see that, compared to Figure 26, outliers are now not considered as part of our clusters, and classified as noise points instead. This prevents our clusters from being affected by them.

The strength of DB Scan is, however, also its weakness. The requirement of two constants, Epsilon and MinPts, means that clustering of all the data must follow this single threshold. This threshold may be more or less appropriate depending on the scarcity of datapoints within the local area. We relook at Fig 20 for an example.

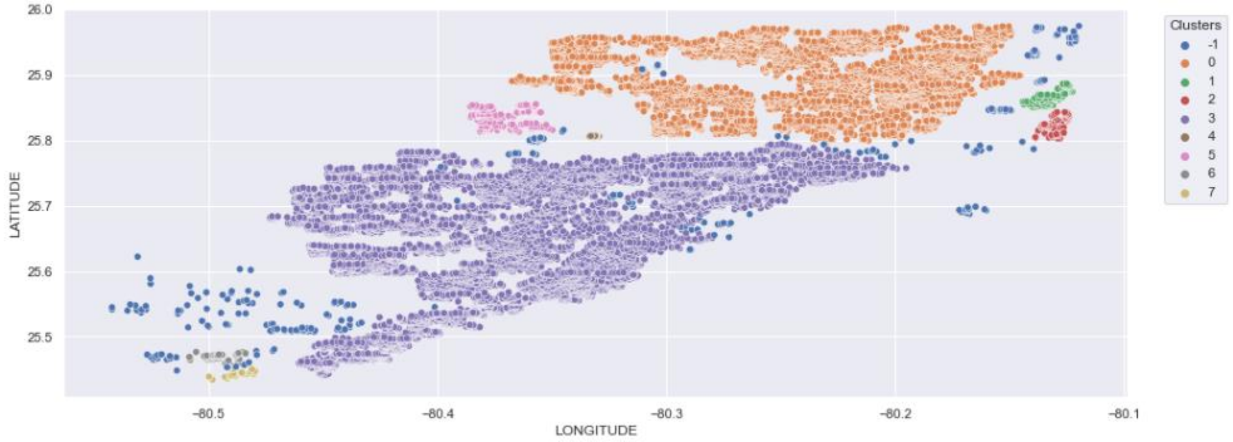


Figure 28: DB Scan with Epsilon = 0.0085, minPts = 20

Although the clustering seems visually appropriate for the most part of this dataset, we realise on closer inspection that almost all of the datapoints on the bottom left is classified as noise points. This is not incorrect based on our selection of Epsilon and minPts. However, if our purpose is to determine where and how many hospitals to build, ideally, we would want to classify those under one cluster as well. That way, one would be built to serve all those within that “outlier cluster”, instead of ignoring them completely. It would thus seem like the threshold should not be an absolute constant, but a value that varies based on a rough estimate of local area density. This could possibly be done by executing a hybrid clustering method – DB Scan would be performed first; For every area comprising a high density of noise points, K-Means or K-Means++ could be performed within that local area.

6 Limitations

The optimal number of clusters for K-means and K-means++ can be determined using many methods like the elbow curve and the silhouette values. Additional methods like selecting initial clusters of K means can also be done via concepts of “valley points” and “peak points”[18].

While DB Scan is able to cluster better than K-means and K-means++ especially when it come to non-spherical data, research also exists on hybrid clustering methods that could potentially better suit certain objectives. Density-based K-means clustering (DBK means) could overcome the shortfalls of both K-means and DB Scan[19]. This could be a potential area for further exploration. However, all said and done, it is impossible to determine a single best clustering method.

7 Conclusion

In conclusion, this project has explored parameters required by K-means, K-means++, and DB Scan and how they may all be determined – optimal K via Elbow Curve and Silhouette Scores, Epsilon via K-NearestNeighbours (with K = square root of data volume), and minPts as 4. Additionally, this project has compared K-means and K-means++ on the how well the latter initializes centroids. Despite the overhead caused by such an initialization, K-means++ generally clusters the data better and should be used over K-means. However, in the rare case where speed is priority, one may choose K-means over K-means++ even though the accuracy maybe lower. K-means and K-means++ however perform badly when it comes to non-spherical data as it is forced to fit the clusters into a given number k. DB Scan was better able to cluster such types of data. However, each method is suited to different types of data. Not only must clustering methods be chosen based on specific use cases, parameters need also to be tweaked. Existing research and “optimal” parameter values can be a mere guide at best – careful consideration must be exercised in deciding the clustering method and values based on unique situations.

8 REFERENCES

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A Review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999, doi: 10.1145/331499.331504.

[2] E. Raff Booz and A. Hamilton, “Exact Acceleration of K-Means++ and K-Means,” *arXiv*, May 2021.

[3] Charles Zaiontz, “Initialize clusters k-means++ | Real Statistics Using Excel,” *real-statistics*, 2021. https://www.real-statistics.com/multivariate-statistics/cluster-analysis/initializing-clusters-k-means/ (accessed Oct. 13, 2022).

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *KDD-96 Proc.*, vol. 2, 1996.

[5] “Elbow Method for optimal value of k in KMeans - GeeksforGeeks,” *GeeksForGeeks.org*, Aug. 22, 2022. https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/ (accessed Oct. 13, 2022).

[6] “2.3. Clustering — scikit-learn 1.1.2 documentation,” *Sci-Kit Learn*, 2022. https://scikit-learn.org/stable/modules/clustering.html (accessed Oct. 13, 2022).

[7] Asanka Perera, “Finding the optimal number of clusters for K-Means through Elbow method using a mathematical approach compared to graphical approach,” *Linkedin*, Oct. 02, 2017. https://www.linkedin.com/pulse/finding-optimal-number-clusters-k-means-through-elbow-asanka-perera (accessed Oct. 15, 2022).

[8] “sklearn.metrics.silhouette_score — scikit-learn 1.1.2 documentation,” 2022. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html (accessed Oct. 16, 2022).

[9] “sklearn.preprocessing.StandardScaler — scikit-learn 1.1.2 documentation,” *Sci-Kit Learn*, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler (accessed Oct. 15, 2022).

[10] “Compare the effect of different scalers on data with outliers — scikit-learn 1.1.2 documentation,” *Sci-Kit Learn*, 2022. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html (accessed Oct. 15, 2022).

[11] Kamil Mysiak, “Explaining DBSCAN Clustering. Using DBSCAN to identify employee… | by Kamil Mysiak | Towards Data Science,” *Towards Data Science*, Jul. 16, 2020. https://towardsdatascience.com/explaining-dbscan-clustering-18eaf5c83b31 (accessed Oct. 15, 2022).

[12] Amey Band, “How to find the optimal value of K in KNN? | by Amey Band | Towards Data Science,” *Towards Data Science* , May 23, 2020. https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb (accessed Oct. 15, 2022).

[13] GeeksForGeeks, “Data Pre-Processing with Sklearn using Standard and Minmax scaler - GeeksforGeeks,” *GeeksForGeeks*, Feb. 03, 2022. https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/?ref=gcse. (accessed Oct. 15, 2022).

[14] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, “Scalable K-Means++,” *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 622–633, Mar. 2012, doi: 10.48550/arxiv.1203.6402.

[15] Jash Sheth, “K+ Means Clustering algorithm,” 2019. https://iq.opengenus.org/k-plus-means-

algorithm/ (accessed Oct. 15, 2022).

[16] Education Ecosystem (LEDU), “Understanding K-means Clustering in Machine Learning | by Education Ecosystem (LEDU) | Towards Data Science,” *Towards Data Science*, Sep. 13, 2018. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (accessed Oct. 15, 2022).

[17] Imad Dabbura, “K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks | by Imad Dabbura | Towards Data Science,” *Towards Data Science* , Sep. 18, 2018. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a (accessed Oct. 15, 2022).

[18] Z. Jiang, “A New Initialization Method for K-means Algorithm Based on Clustering Coefficient,” *J. Phys. Conf. Ser.*, vol. 1992, no. 4, pp. 0–7, 2021, doi: 10.1088/1742-6596/1992/4/042060.

[19] S. Chakraborty, N. K. Nagwani, and L. Dey, “Performance Comparison of Incremental Kmeans and Incremental DBSCAN Algorithms,” *Int. J. Comput. Appl.*, vol. 27, no. 11, pp. 14–18, 2011, doi: 10.5120/3346-4611.