NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4032: Data Analytics and Mining

Group 27

Name	Matriculation Number	Methods
SHALINA SHAM @LIM SHANA	U2022851K	Hierarchical Agglomerative
FU XIAOXUAN	U2020847G	Hierarchical Agglomerative
LIM YUN HAN, DARREN	U1921275J	Fuzzy C Means
YI JIA XIN, JOCELINE	U1920057J	K-Means
LUMLERTLUKSANACHAI PONGPAKIN	U2023344C	DBSCAN

Abstract

This paper aims to compare the performance of clustering algorithms on two separate datasets – the Iris and Wine dataset. To do so, we employed 4 different common clustering algorithms and tested them over the 2 known datasets. We varied several key parameters for each clustering method in order to define an optimal model for each method. Comparison was done across all clustering methods using 3 performance metrics as the standard of evaluation. The results affirm that different clustering methods have different performance standards for each dataset.

1 Introduction

Clustering is a powerful machine learning tool used for detecting structures in datasets and can refer to the process of grouping unlabelled data points. Unlike supervised methods in machine learning, clustering is considered an unsupervised method that works well on datasets with no target variables where knowledge of the relationship between the data points is unknown.

This machine learning method, though relatively simple in its concept, has a myriad of applications which include market segmentation, social network analysis, image segmentation, and anomaly detection.

In this paper, we will be performing a comprehensive analysis of 4 different clustering methods commonly used. These methods are Agglomerative Hierarchical Clustering, Fuzzy C Means, K-Means, and DBSCAN.

2 Related Work

A variety of approaches have been used to compare and analyze the performance of various clustering algorithms based on various datasets. The approaches differ between comparing across various clustering algorithms and comparing different types of the same algorithm.

A comparative analysis of different clustering algorithms is presented in several works. Shah et al. [1] analyzed the performance of 6 clustering algorithms among 6 datasets, the algorithms being K-Means clustering, hierarchical clustering EM clustering, density-based clustering, farthest first clustering and canopy clustering. It was concluded the K-Means had the best performance, whereas EM algorithm had the worst performance. Nerurkar et al. [2] also reviewed other clustering algorithms using 2 artificially generated datasets. Algorithms reviewed include K-Means, K-Means++, Kernel K-Means, Hierarchical clustering, Fuzzy C-Means, model based, DBSCAN and OPTICS, and various drawbacks were identified such as sensitivity of outliers and noise. Gelbard et al. [3] analyzed 10 clustering algorithms and tested over 4 datasets and tried to explain the differences between the clustering results.

Various works have also conducted in-depth analysis for different types of a single clustering method. Gosain et al. [4] and Döring et al. [5] reviewed the performance of all major fuzzy clustering algorithms on datasets in the presence of noise and outliers. Cecil et al. [6] reviewed and analyzed hierarchical clustering methods over different attributes such as linkage types. Burghardt et al. [7] reviewed and analyzed agglomerative and divisive hierarchical clustering.

3 Metrics

We will be employing four clustering methods on two datasets – the ‘Iris’ [8] and ‘Wine’ [9] datasets, both taken from the UCI Machine Learning Repository on Kaggle. The ‘Iris’ dataset is a multivariate dataset with 150 data points consisting of 4 features (Sepal Length, Sepal Width, Petal Length, and Petal Width) that define the species of the iris plant (specifically Iris Setosa, Iris Versicolor, and Iris Virginica). The ‘Wine’ dataset consists of multivariate data from the results of a chemical analysis of wine grown in Italy but derived from 3 different cultivars. There are 13 constituents found in each of the three types of wines, making up the features of the wine. We selected these datasets as they are the best-known datasets used in many classic machine learning applications. The small size of the dataset also allows us to cluster the points in a more efficient way and allow us to better analyse the clustering methods.

For the analysis of the clustering methods, we will be using 8 performance metrics to determine how well the clusters have been formed by their respective algorithms. The metrics used are as follows: Silhouette Score, Davies Bouldin Score, Calinski-Harabasz Score, V Measure Score, Adjusted Random Score, Adjusted Mutual Information and Accuracy.

3.1 Silhouette Score

The silhouette score/coefficient [10] is used to calculate the goodness of the clustering technique by measuring the separation distance between resulting clusters. The silhouette score for a cluster is defined by the formula $s = \frac{n-1}{\max(i.n)}$, where *i* is the mean distance within each cluster and *n* is the distance between each sample and the nearest neighbouring cluster and has a range of [-1,1].

A score nearing +1 is a 'good' score as it indicates that the clusters are far away from the neighbouring clusters and clearly distinguished. A score of 0 shows that the distance between clusters is not significant and negative values show that the samples might have been assigned to the wrong cluster. Averaging the Silhouette Coefficients allows us to obtain a global Silhouette Score which can then be used to describe the entire population's performance.

Silhouette plots can also be used to display how close each point in one cluster is to other points in neighbouring clusters, allowing us to assess parameters like number of clusters visually. The thickness of a silhouette plot can be used to determine the cluster size visually as well.

3.2 Davies Bouldin Score

The Davies Bouldin (DB) score/index [11] is used to evaluate the goodness of split and can be defined as the average similarity of each cluster with a cluster most similar to it. In order to calculate the DB index, the intra-cluster dispersion (average distance between each observation and its centroid), separation measure (separation between two clusters *i* and *j*) and similarity between clusters has to be precomputed first. After which the most similar cluster for each cluster *i* is determined and the DB index can be calculated using the formula $\bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$.

As similarity is measured as the ratio of within-cluster distances to between-cluster distances, clusters which are farther apart and less dispersed will lead to a better score. A 'good' cluster can be defined as one where its average similarity is minimized, hence a lower DB score (minimum score = 0) indicates a better cluster.

3.3 Calinski-Harabasz Score

The Calinski-Harabasz (CH) score/index [12] (also known as Variance ratio criterion) can be used to evaluate a clustering model when the ground truth labels of a dataset are not known. Validation of how well the clustering has performed using the CH score is made using quantities and features inherent to the dataset. CH score is a measure of how similar an object is to its own cluster compared to other clusters. It uses the concept of cohesion and separation, where cohesion is estimated based on the distances from the data points in a cluster to its own cluster centroid and separation is based on the distance of the cluster centroids from the global centroid. The CH index for a *k* number of clusters on a dataset D = $[d_1, d_2, \ldots, d_N]$ can be defined as

$$CH = \frac{\left[\frac{\sum_{k=1}^{K} n_k \|c_k - c\|^2}{K-1}\right]}{\left[\frac{\sum_{k=1}^{K}\sum_{i=1}^{n_k} \|d_i - c_k\|^2}{N-K}\right]}$$

, where n_k and c_k are the number of points and centroid of the k^{th} cluster respectively, *c* is the global centroid and *N* is the total number of data points. Higher values of CH index means that the clusters are dense and well-separated.

3.4 V Measure Score

The V Measure score (also known as Normalised Mutual Information score) [13] is used to evaluate the clustering algorithm's performance and can be defined as the weighted average of homogeneity and completeness. Homogeneity [14] is used to determine if the clustering method produces clusters which are homogenous, where homogeneity can be defined as clusters that contain only data points which are members of a single class and calculated by the formula $h = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})}$. The completeness score [15] is used to compute the completeness metric of a

cluster labelling given a ground truth. A clustering satisfies completeness if all the data points that are members of the same given class are also elements of the same cluster. Completeness score can be computed by the formula $c = 1 - \frac{H(Y_{pred}|Y_{true})}{H(Y_{pred})}$ and is symmetric to the homogeneity score as they are complementary. Precomputation of homogeneity and completeness is required, thereafter the V measure score can be computed with the following formula: $v = \frac{(1+\beta) \times homogeneity \times completeness}{\beta \times homogeneity \times completeness}$, where β is defaulted to 0. Using a β value higher than 1 indicates that more weight is attributed to homogeneity and β value of less than 1 indicates that more weight is attributed to completeness. The V measure score is also bounded and given in the range of [0,1], much like the completeness and homogeneity score and a score of 1 indicates perfectly complete and homogenous labelling whilst a score of 0 indicates non-complete and homogenous labelling.

3.5 Adjusted Random Score

The adjusted random score/index (ARI) [16] is the raw random index score adjusted for chance. The random index score computes the similarity between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusters. By adjusting the random index, the ARI is now ensured to have a value close to 0 for random labelling independently of the number of clusters and samples and exactly 1 when the clusters are identical. The ARI can be computed with the formula: $ARI = \frac{RI - expected\ RI}{Max(RI) - expected\ RI}$, where RI is a precomputed value.

3.6 Adjusted Mutual Information Score

The adjusted mutual information score (AMI) [17] is an adjustment of the Mutual Information (MI) score to account for chance, where MI is a measure of the similarity between 2 labels of the same data. The AMI accounts for the fact the MI is generally higher for 2 clusterings with a larger number of clusters, regardless of whether there is more information shared in actuality. The AMI is given by: $AMI(U,V) = \frac{MI(U.V) - E(MI(U,V))}{avg(H(U),H(V)) - E(MI(U,V))}$. A 'good' AMI score is given by 1, where it represents perfect labels that are homogenous and complete, and a score of 0 denotes class members being split across different clusters.

3.7 Accuracy

Accuracy [18] can also be used to judge a machine learning algorithm. Accuracy is defined as a fraction of the total number of correctly predicted data points out of the total number of data points or the total number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives. A true positive or true negative is a data point that the algorithm correctly classified as true or false, respectively. A false positive or false negative, on the other hand, is a data point that the algorithm incorrectly classified.

4 Methods

4.1 Agglomerative Hierarchical Clustering

Hierarchical clustering is a clustering technique that seeks to build a hierarchy of clusters. There are two main strategies - agglomerative and divisive. Agglomerative hierarchical clustering is a bottom-up approach, where each point is an individual cluster, and with every iteration, the closest pair of clusters is merged until only a single cluster remains or the target number of clusters is achieved. Divisive hierarchical clustering is a top-down approach, where all data points begin in the same cluster, and are recursively split into different clusters. For the purpose of this paper, we will be focusing on hierarchical agglomerative clustering.

In hierarchical agglomerative clustering, we start by taking every data point as a cluster. Then two clusters are merged with every iteration. In order to decide which two clusters to merge, the pairwise distance between any two clusters is compared and the pair with the minimum pairwise distance is selected. There are various ways to determine the pairwise distance between two clusters, which will be elaborated in the following sections.

A dendrogram (binary clustering tree) can be used to visualize the merging process when clustering is performed. The dendrogram

represents a hierarchy of partitions, whose root is the class that contains all the data points and each node on the tree is a cluster.

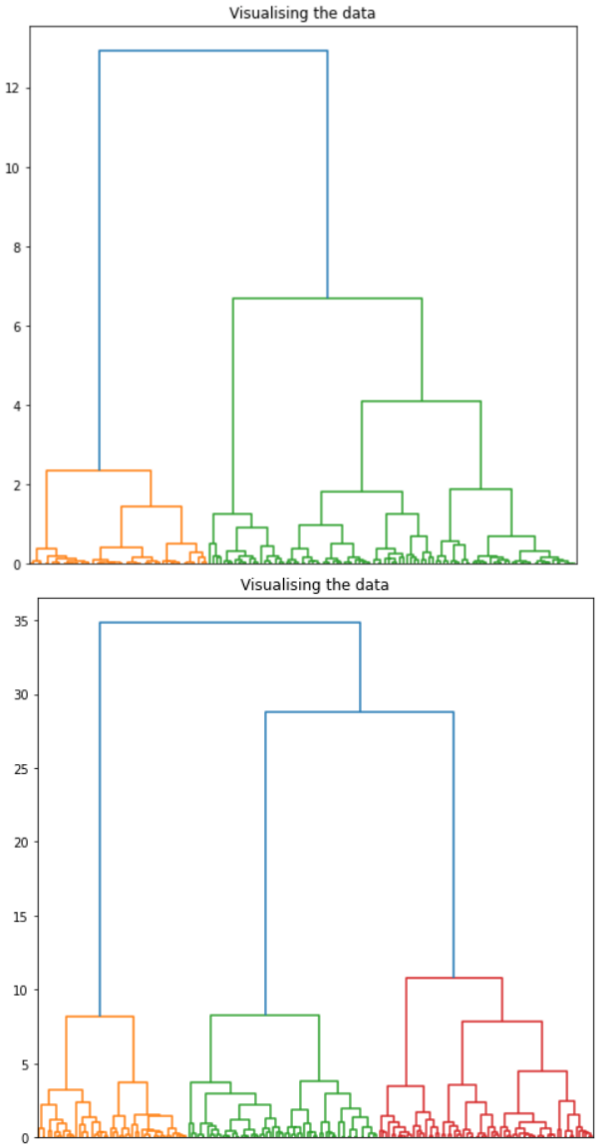

***Figure 1**: Dendrogram generated from Iris dataset (top) and Wine dataset (bottom)*

Strengths:

1. The algorithm is simple to understand and implement.
2. There is no need to assume any particular number of clusters.
3. The algorithm outputs a hierarchy, a structure that is more informative which makes it easier to decide on the number of clusters by looking at the dendrogram.

Weaknesses:

1. Once two clusters are combined, the action cannot be undone.
2. The algorithm is sensitive to noise and outliers.
3. There is difficulty handling different sized clusters and irregular shapes.
4. The order of the data impacts the final results of the clustering.
5. It is not suitable for large datasets due to high time and space complexity.

4.1.1 Key Factors Affecting Performance

Time Complexity

Hierarchical agglomerative clustering requires performing *n* iterations and, in each iteration, we need to update the similarity matrix and restore the matrix, resulting in a costly time complexity. The time complexity is the order of the cube of *n*, $O(n^3)$ where *n* is the number of data points.

Space Complexity

The space required for hierarchical agglomerative clustering is very high when the number of data points is high as we need to store the similarity matrix in the Random Access Memory (RAM). The space complexity is the order of the square of *n*, $O(n^2)$, where *n* is the number of data points.

Hence, one of the most important factors affecting the performance of hierarchical agglomerative clustering is the size of the dataset as it tends to perform poorly with larger datasets.

4.1.2 Experiments with Various Parameters

Number of Clusters

The number of clusters in agglomerative hierarchical clustering do not have to be predetermined, and there is no standard measure to choose the number of clusters. Hence, the number of clusters are chosen based on the performance metrics described as well as prior knowledge on the dataset used. We decided to explore the number of clusters for a range of 2-7 and determine the optimal value of *k* based on the performance of the algorithm against the metrics. This range was chosen as a standard across the rest of the algorithms mentioned. However, it can be noted that hierarchical agglomerative

clustering allows for the maximum number of clusters to be the total number of data points in a dataset since each point starts off as a cluster in the initial stages of hierarchical agglomerative clustering [19].

Clusters	2	3	4	5	6	7
Accuracy	0.66	0.78	0.79	0.79	0.81	0.89
Silhouette Score	0.63	0.61	0.57	0.51	0.50	0.50
Davies-Bouldin Score	0.51	0.47	0.60	0.61	0.57	0.69
Calinski-Harabasz Score Score	289.24	380.24	462.58	458.28	456.99	493.70
V Measure Score	0.69	0.66	0.60	0.60	0.65	0.72
ARI Score	0.54	0.57	0.55	0.55	0.59	0.72
AMI Score	0.69	0.66	0.60	0.60	0.65	0.71

Table 1: *Results from hierarchical agglomerative clustering for number of clusters (Iris dataset)*

The most optimal results for each performance metric have been highlighted in Table 1. According to the Silhouette Score, the optimal number of clusters is 2 as it has the highest score compared to the rest of the cluster values. The DB score, however, indicates that the optimal number of clusters is 3 as it has the lowest score. On the other hand, accuracy, CH, V measure, ARI and AMI scores favor 7 clusters as 7 clusters produced the best results. Based on the results in Table 1 and prior knowledge of the Iris dataset, we have concluded on a cluster value of 3 as the algorithm produced decent performance metric scores when 3 clusters are used.

Clusters	2	3	4	5	6	7
Accuracy	0.66	0.97	0.97	0.97	0.97	0.97
Silhouette Score	0.47	0.56	0.48	0.39	0.38	0.40
DB Score	0.73	0.60	0.72	0.84	0.85	0.84
CH Score	157.72	341.06	307.24	283.56	289.89	319.69
V Measure Score	0.63	0.86	0.86	0.86	0.86	0.86
ARI Score	0.63	0.86	0.86	0.86	0.86	0.86
AMI Score	0.47	0.90	0.90	0.90	0.90	0.90

Table 2: *Results from hierarchical agglomerative clustering for number of clusters (Wine dataset)*

According to accuracy, V Measure, ARI and AMI scores, the most optimal number of clusters to have for the Wine Dataset is 3-7. The CH, silhouette and DB scores indicate that the optimal number of clusters should be 3 for the Wine dataset. We have thus decided to select having 3 clusters as the optimal since most of the parameters agree that the optimal results are achieved when 3 clusters are used.

From these experiments, the determined optimal number of 3 clusters for both the Iris and Wine dataset will be used in the experiments that follow.

Linkage Types

At each step of the agglomerative hierarchical clustering, the 2 clusters separated by the shortest distance are combined. The function used to determine the distance between two clusters, is known as the linkage function. There are many linkage types used in clustering algorithms, but we will only be focusing on single, average, complete and ward linkage [20] in this paper.

Single linkage returns the minimum distance between 2 points, where each point belongs to 2 different clusters. The benefit of this function is that it can deal with non-standard shapes of clusters but cannot separate clusters properly if there is noise.

Average linkage returns the average of distances between all pairs of data points. It is able to separate clusters if there is noise between the clusters but is biased towards clusters that form blots.

Complete linkage returns the maximum distance between elements in clusters. This does well in separating clusters if there is noise between them,

but it is biased towards circular clusters and tends to break up large clusters.

Ward linkage returns the increase in the error sum of squares (ESS) after fusing 2 clusters into a single cluster. It seeks to choose the successive clustering steps so as to minimize the increase in ESS at each step. Ward linkage is also able to separate clusters if there is noise and is also biased towards circular groups of clusters, much like complete linkage.

We have opted to use Euclidean affinity for testing of all linkage types since any other type of affinity when used with respect to the ward linkage is inadequate and can only be partly justified [21].

Linkage Type	Ward	Complete	Average	Single
Accuracy	0.78	0.85	0.85	0.67
Silhouette Score	0.61	0.63	0.63	0.48
Davies-Bouldin Score	0.47	0.54	0.54	0.46
Calinski-Harabasz Score	380.24	459.25	459.25	154.27
V Measure Score	0.66	0.70	0.70	0.73
ARI Score	0.57	0.65	0.65	0.57
AMI Score	0.66	0.70	0.70	0.73

Table 3: *Results from hierarchical agglomerative clustering for Linkage Type (Iris dataset)*

Accuracy, silhouette, CH, and ARI scores favor either complete or average linkage while DB, V Measure and AMI scores suggest that single linkage produces the most optimal results. Based on this, we have decided that using either complete or average linkage for hierarchical agglomerative clustering on the Iris dataset will be the most ideal as most metrics agree with this conclusion. For the experiments that follow, we have decided to choose the usage of complete linkage for the Iris dataset.

Linkage Type	Ward	Complete	Average	Single
Accuracy	0.97	0.87	0.92	0.41
Silhouette Score	0.56	0.51	0.55	0.02
Davies-Bouldin Score	0.60	0.63	0.60	0.63
Calinski-Harabasz Score	341.06	277.35	333.53	2.58
V Measure Score	0.86	0.70	0.78	0.02
ARI Score	0.90	0.66	0.77	0.00
AMI Score	0.86	0.70	0.78	0.02

Table 4: *Results from hierarchical agglomerative clustering for Linkage Type (Wine dataset)*

It can be observed that the most optimal linkage type to employ will be ward linkage as it provides the best performance metric scores. However, it is interesting to note that the DB score favors the use of average linkage as well. Therefore, it can be concluded that the best linkage type to use for the task of clustering the Wine dataset is ward linkage.

While the most optimal linkage type for hierarchically clustering the Wine dataset is ward linkage, there is inadequate justification [21] to use affinity types that are not Euclidean with ward linkage. We have hence opted to use of average linkage instead to evaluate the effectiveness of the various affinity types in the next section since it has the next best score values across all metrics.

Affinity Types

The final clusters created by agglomerative hierarchical clustering are also influenced by the affinity metric chosen, mainly how the distance between points is calculated. We will be focusing on the following affinity types: Euclidean (l2), Manhattan (l1) and Cosine [20] in this paper.

Euclidean distance is the default distance metric used to measure the distance between clusters and is the straight-line distance between 2 points and is mathematically expressed as: $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$.

Manhattan distance is the sum of the absolute difference between points across all the dimensions and works as if there was a grid like path between the points. This distance is often

good for sparse features and is mathematically expressed as: $d(x, y) = \sum_{n=1}|x_i - y_i|$.
The Cosine distance metric measures the degree of angle between two vectors and is used when the magnitude between points does not matter but the orientation does. It is often used in natural language programming and is measured as: $\cos(\theta) = \frac{A \cdot B}{\|A\| * \|B\|}$.

Affinity Type	Euclidean	Manhattan	Cosine
Accuracy	0.85	0.78	0.82
Silhouette Score	0.63	0.61	0.62
Davies-Bouldin Score	0.54	0.47	0.57
Calinski-Harabasz Score	459.25	380.24	452.94
V Measure Score	0.70	0.66	0.66
ARI Score	0.65	0.57	0.60
AMI Score	0.70	0.66	0.65

Table 5: *Results from hierarchical agglomerative clustering for Affinity Type (Iris dataset)*

From the results, it is clear to see that Euclidean distance is the most optimal affinity type to employ as it produces the best performance metric scores for all except for DB score. Hence, we will use the Euclidean distance as the affinity type for hierarchical agglomerative clustering of the Iris dataset.

Affinity Type	Euclidean	Manhattan	Cosine
Accuracy	0.92	0.92	0.87
Silhouette Score	0.55	0.55	0.51
Davies-Bouldin Score	0.60	0.60	0.63
Calinski-Harabasz Score	333.53	332.18	277.35
V Measure Score	0.78	0.77	0.70
ARI Score	0.77	0.75	0.66
AMI Score	0.78	0.77	0.70

Table 6: *Results from hierarchical agglomerative clustering for Affinity Type (Wine dataset)*

Accuracy, silhouette and DB scores favor the use of either Euclidean or Manhattan distance as the most optimal affinity type. The remaining metrics all favor the use of Euclidean distance to produce the best results. Hence, the combination of all the scores seem to suggest that Euclidean distance would be the most optimal affinity type to use for hierarchical agglomerative clustering on the Wine dataset.

4.1.3 Results of Hierarchical Agglomerative Clustering

Based on our experiments, it can be noted that the optimal number of centroids for the Iris and Wine dataset is 3, and the optimal linkage type used to cluster the Iris dataset is complete linkage. The optimal linkage type used to cluster the Wine dataset is ward linkage. The optimal affinity type used to cluster both datasets is Euclidean affinity. For the optimal model employed to cluster the Iris dataset, we will be using the number of clusters set to 3, complete linkage, and Euclidean affinity. For the optimal model used to cluster the Wine dataset, we will be using the number of clusters set to 3, ward linkage, and Euclidean affinity.

Iris Dataset

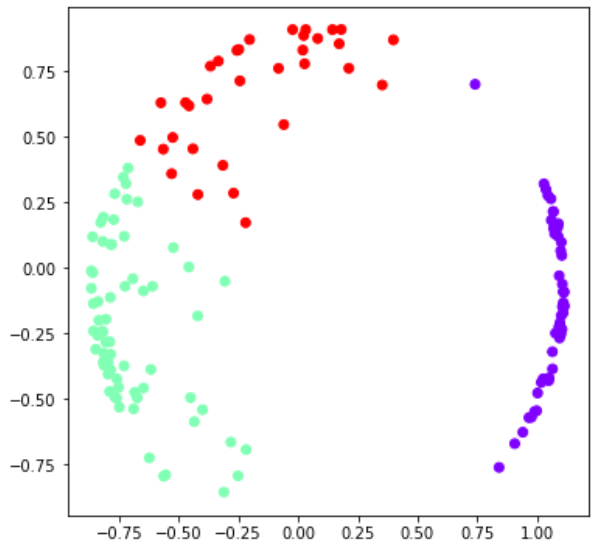

Figure 2*: Scatter plot for hierarchical agglomerative clustering of Iris dataset*

Wine Dataset

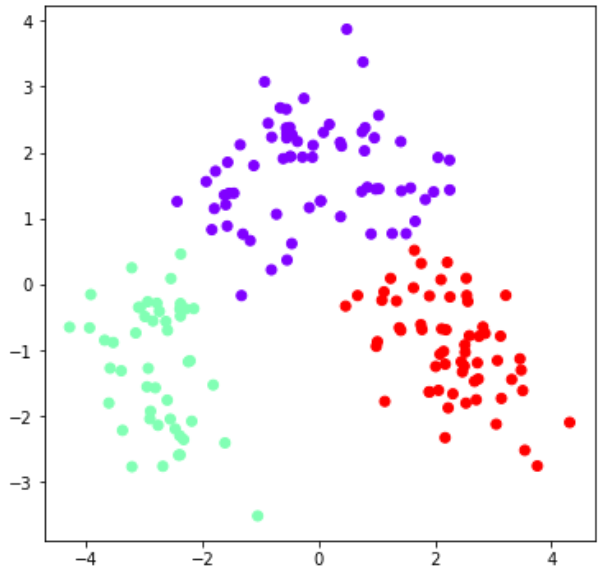

***Figure 3**: Scatter plot for hierarchical agglomerative clustering of Wine dataset*

4.2 Fuzzy C Means

Fuzzy C Means (FCM), also known as soft K means, is similar to the well-known K-means clustering algorithm as it is objectively computed the same way as K-means, but with a new variable introduced – a membership value. This is the biggest difference between Fuzzy C Means and K-means. K-means is a hard clustering algorithm while Fuzzy C Means is a soft clustering algorithm. Hard clustering performs its task in a binary fashion – data points are either completely in a cluster, or not. Soft clustering, on the other hand, assigns a value or probability of a data point belonging in a cluster.

The objective of FCM is to compute membership values to minimize the within-cluster distances and maximize the between-cluster distances. The result of FCM is a membership map for each category. The membership value ranges from 0 to 1, with higher values indicating that a data point more likely belongs to a particular category.

Strengths:

1. Gives best result for overlapped data set and comparatively better than K-means algorithm.
2. Unlike K-means where data point must exclusively belong to one cluster center here data point is assigned membership to each cluster center as a result of which data point may belong to more than one cluster center.
3. Comparatively fast computation

Weaknesses:

1. There is a need for users to specify the number of clusters beforehand
2. It is sensitive to noise and outliers
3. There is difficulty handling different sized clusters and irregular shapes.
4. Clusters are sensitive to the initial cluster assignment

4.2.1 Experiments with Various Parameters

Number of clusters

We experimented with the initialization of different number of clusters to see how well the Fuzzy C Means algorithm will fare, using the performance metrics described as the method of evaluation.

Clusters	2	3	4	5	6	7
Accuracy	0.65	0.89	0.69	0.54	0.51	0.45
Silhouette Score	0.68	0.55	0.50	0.37	0.37	0.33
Davies-Bouldin Score	0.40	0.67	0.77	0.91	0.92	1.02
Calinski-Harabasz Score Score	513.30	560.37	528.86	457.96	474.56	437.25
V Measure Score	0.66	0.74	0.70	0.65	0.63	0.60
ARI Score	0.54	0.72	0.61	0.46	0.45	0.40
AMI Score	0.65	0.74	0.70	0.64	0.64	0.61

***Table 7:** Results from Fuzzy C Means for number of clusters (Iris dataset)*

As seen from Table 7, the optimal number of clusters for the Iris dataset is 3, as it maximizes the accuracy, CH, V measure, ARI, and AMI scores. However, it is interesting to note that the silhouette score and DB score favors a smaller number of clusters.

Clusters	2	3	4	5	6	7
Accuracy	0.66	0.69	0.59	0.56	0.42	0.38
Silhouette Score	0.66	0.57	0.56	0.56	0.54	0.51
Davies-Bouldin Score	0.48	0.54	0.55	0.51	0.52	0.57
Calinski-Harabasz Score Score	505.13	559.70	705.55	725.25	853.79	816.95
V Measure Score	0.41	0.42	0.38	0.36	0.39	0.36
ARI Score	0.36	0.36	0.29	0.35	0.24	0.22
AMI Score	0.40	0.41	0.37	0.27	0.37	0.34

Table 8: *Results from Fuzzy C Means for number of clusters (Wine dataset)*

From the results, it can be seen that the optimal number of clusters for the Wine dataset is 3, as it maximizes the accuracy, V measure, ARI, and AMI score. However, it can be noted that a cluster number of 2 also maximizes the silhouette, DB, and ARI score. It is also interesting to note that the CH score performs the best when number of clusters is 6.

As determined by the results, the optimal number of clusters for Fuzzy C Means is 3 for both the Iris and Wine dataset, and we will be using this value in the subsequent experiment.

Initialization

The initial initialization of the cluster is a parameter that can affect the outcome of the model as well. To see how the performance varies for different initial cluster initialization, we ran the model on the Iris and Wine dataset and noted the metrics for each random initialization. We first performed ran the model against random cluster initialization, then ran the model with a different cluster initialization – initialization of cluster centers at random locations within a multi-variate Gaussian distribution with zero-mean and unit-variance.

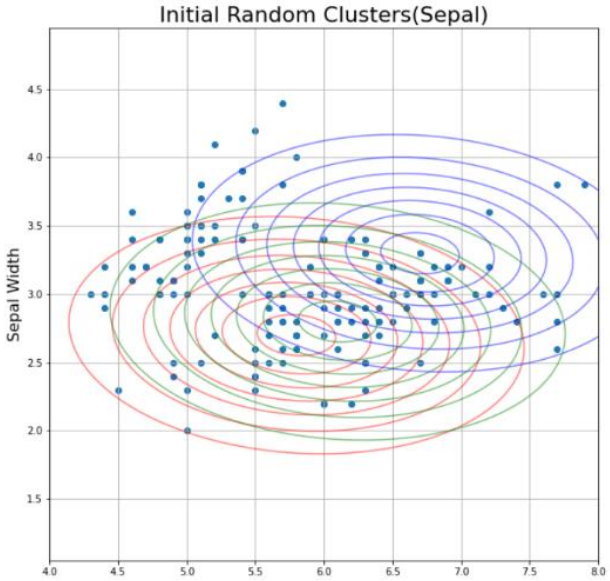

Figure 4*: Random Initial Cluster - 1st iteration (Iris dataset)*

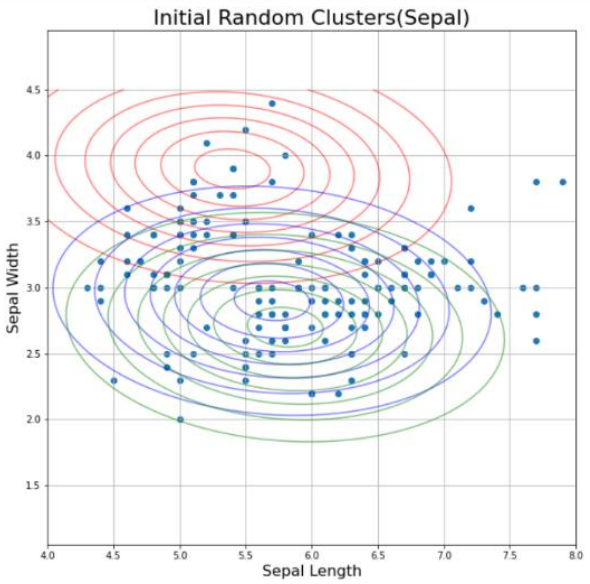

Figure 5*: Random Initial Cluster (2nd iteration)*

Iteration	1	2	3	4	5
Accuracy	0.89	0.89	0.89	0.89	0.89
Silhouette Score	0.55	0.55	0.55	0.55	0.55
Davies-Bouldin Score	0.67	0.67	0.67	0.67	0.67
Calinski-Harabasz Score	560.37	560.37	560.37	560.37	560.37
V Measure Score	0.74	0.74	0.74	0.74	0.74

ARI Score	0.72	0.72	0.72	0.72	0.72
AMI Score	0.74	0.74	0.74	0.74	0.74

Table 9: *Results from Fuzzy C Means for random cluster initialization (Iris dataset)*

As noted from Table 9, there is no change in any values for the performance metrics for all 5 iterations of the model. This may be due to the small size of the data, making it less complex and with less noise. We then ran the model against the second initialization – one with multi-variate Gaussian distribution.

Iteration	**1**	**2**	**3**	**4**	**5**
Accuracy	0.89	0.89	0.89	0.89	0.89
Silhouette Score	0.55	0.55	0.55	0.55	0.55
Davies-Bouldin Score	0.67	0.67	0.67	0.67	0.67
Calinski-Harabasz Score	560.37	560.37	560.37	560.37	560.37
V Measure Score	0.74	0.74	0.74	0.74	0.74
ARI Score	0.72	0.72	0.72	0.72	0.72
AMI Score	0.74	0.74	0.74	0.74	0.74

Table 10: *Results from Fuzzy C Means for Gaussian distribution cluster initialization (Iris dataset)*

Comparing Table 9 and 10, it can be observed that there is no difference in the performance of the model with the 2 different cluster initialization methods for the Iris dataset.

Iteration	**1**	**2**	**3**	**4**	**5**
Accuracy	0.69	0.69	0.69	0.69	0.69
Silhouette Score	0.57	0.57	0.57	0.57	0.57
Davies-Bouldin Score	0.54	0.54	0.54	0.54	0.54
Calinski-Harabasz Score	559.70	559.70	559.70	559.70	559.70
V Measure Score	0.42	0.42	0.42	0.42	0.42
ARI Score	0.36	0.36	0.36	0.36	0.36
AMI Score	0.41	0.41	0.41	0.41	0.41

Table 11: *Results from Fuzzy C Means for random cluster initialization (Wine dataset)*

Running the model with random cluster initialization for the Wine dataset, we also note the same results as the Iris dataset. There is no change in values for all performance metrics for all 5 iterations.

Iteration	**1**	**2**	**3**	**4**	**5**
Accuracy	0.69	0.69	0.69	0.69	0.69
Silhouette Score	0.57	0.57	0.57	0.57	0.57
Davies-Bouldin Score	0.54	0.54	0.54	0.54	0.54
Calinski-Harabasz Score	559.70	559.70	559.70	559.70	559.70
V Measure Score	0.42	0.42	0.42	0.42	0.42
ARI Score	0.36	0.36	0.36	0.36	0.36
AMI Score	0.41	0.41	0.41	0.41	0.41

Table 12: *Results from Fuzzy C Means for Gaussian distribution cluster initialization (Wine dataset)*

Comparing Table 11 and 12, we can observe that there is also no difference in the performance of the model with the 2 different cluster initialization methods for the Wine dataset.

From the experiments done above, we can conclude that while cluster initialization is an important factor in clustering, it does not affect our FCM model for both the Iris and Wine dataset.

4.2.2 Results from Fuzzy C Means clustering

Based on our experiments, it can be noted that there is no significant difference in the initialization of centroids, and the optimal

number of centroids for the Iris and Wine dataset is 3. For the optimal model employed to cluster both datasets, we will be using the random centroid initialization with number of clusters set to 3.

Iris dataset

Number of Clusters	3
Initialisation	**Random**
Accuracy	0.89
Silhouette Score	0.55
Davies-Bouldin Score	0.67
Calinski-Harabasz Score	560.37
V Measure Score	0.74
ARI Score	0.72
AMI Score	0.74

Table 13: *Results from Fuzzy C Means for Iris dataset*

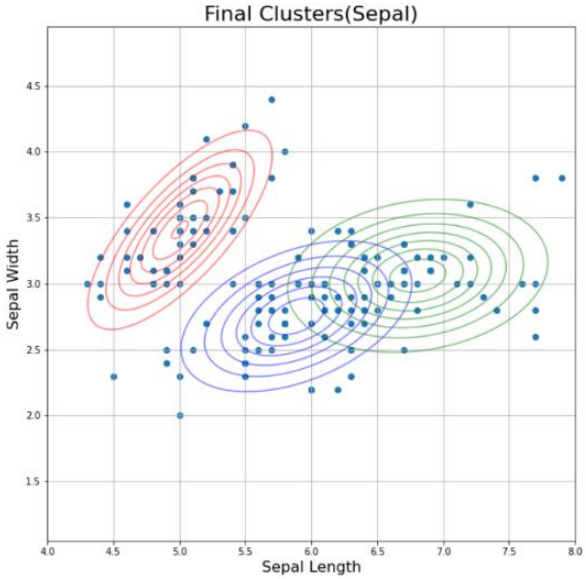

Figure 6*: Fuzzy C Means clustering model on Iris dataset (Sepal Features)*

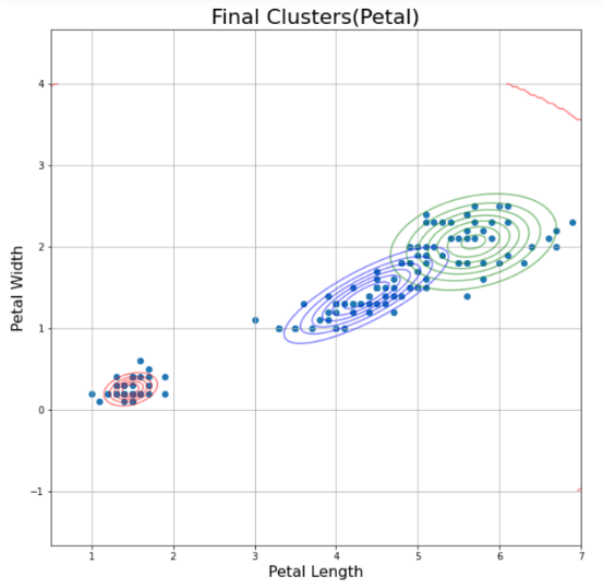

Figure 7*: Fuzzy C Means clustering model on Iris dataset (Petal Features)*

4.3 K-Means Clustering

K-means clustering is a centroid-based algorithm that aims to minimize the sum of distances between the points in a cluster and their cluster centroid. The first step of clustering using this method would be to determine the number of clusters, *k*. To do so, the elbow method was employed. The elbow method runs K-means clustering on the dataset for a range of values of *k* (in this case, the range 1-7 was used). For each value of *k*, the sum of squared distance between each point and the centroid in a cluster (WCSS or Within-Cluster Sum of Squares) is computed and plotted on a line graph. To determine *k*, the value at the 'elbow' is selected.

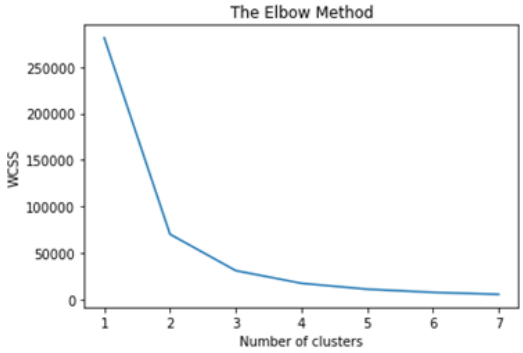

***Figure 8**: Elbow method performed on Iris dataset*

The next step of K-means would be to randomly initialize *k* cluster centroids and then initiate a loop with an assignment step and a centroid update step. The assignment step assigns each point to its closest centroid by calculating the Euclidean distance and the centroid update step will compute the new centroid of each cluster. The loop will run until the position of the centroids do not change.

Strengths:

1. K-means is relatively simple to implement
2. Ability to scale K-means to large datasets
3. Guarantees convergence

Weakness:

1. Number of cluster centroids, *k*, has to be chosen manually using the elbow method.
2. K-means is sensitive to outliers, which means outliers may either get their own cluster or centroids can be dragged by outliers.
3. K-means cannot work well on data with non-spherical shapes.
4. K-means is also sensitive to the initialization of cluster centroids. A poor initialization of the cluster centroids may lead to different results.

4.3.1 Experiments with Various Parameters

Number of clusters

The number of cluster centroids was initially determined by the elbow method as mentioned. However, we also varied the number of clusters and ran the k-means model against the variation to determine the optimal number of clusters based on the performance metrics mentioned.

Clusters	2	3	4	5	6	7
Accuracy	0.65	0.89	0.01	0.09	0.50	0.44
Silhouette Score	0.68	0.55	0.50	0.49	0.37	0.36
Davies-Bouldin Score	0.40	0.66	0.78	0.82	0.92	0.97
Calinski-Harabasz Score Score	513.30	560.39	529.12	493.92	474.42	450.41
V Measure Score	0.66	0.75	0.72	0.70	0.64	0.67
ARI Score	0.54	0.73	0.65	0.61	0.45	0.47
AMI Score	0.65	0.75	0.72	0.69	0.63	0.66

***Table 14:** Results from K-Means for number of clusters (Iris dataset)*

For the Iris dataset, we can determine that the optimal number of clusters is 3 (as it has the highest values for the majority of the metrics), which concurs with the initial determination using the elbow method. It is interesting to note that the clusters are more homogenously and completely labelled (based on V measure, ARI, and AMI score) when number of clusters is set to 3 but does not fare so well when using metrics such as silhouette score and DB score.

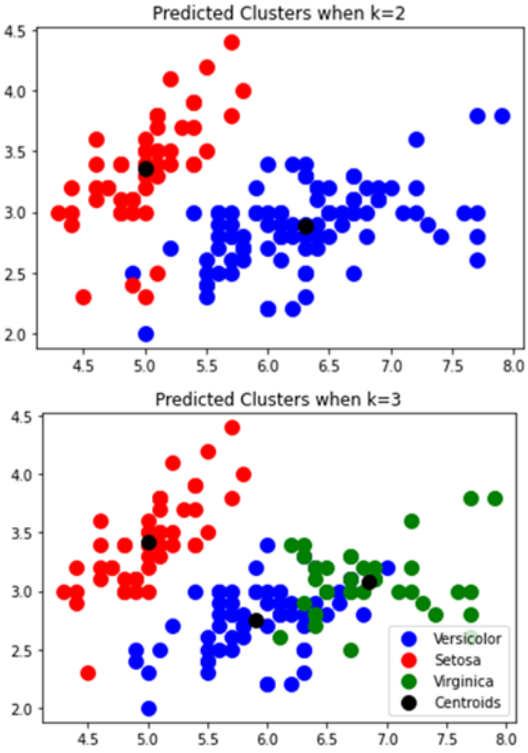

***Figure 9**: Predicted clusters when k=2 (top) and k=3 (bottom) for Iris dataset*

A plausible reason for the observation of why a higher number of clusters for the Iris dataset does not fare so well at the silhouette and DB score can be visually explained in the clustering shown above when *k* is set to 2 and 3 respectively. Silhouette score calculates the separation distance of the clusters whilst DB score calculates the average similarity of the cluster with another cluster most similar to it. When *k* is set to 2, the centroids of the clusters are more far apart, showing that the clusters are further away from their neighbouring cluster and more distinguished, leading to a better silhouette score. As the clusters are farther apart and less dispersed, it results in a better DB score. On the other hand, when *k* is set to 3, the centroids of the clusters are nearer, resulting in nearer clusters and less distinguished ones, thus explaining the lower performing silhouette and DB score when *k* is set to 3.

Clusters	**2**	**3**	**4**	**5**	**6**	**7**
Accuracy	0.54	0.70	0.58	0.21	0.53	0.13
Silhouette Score	0.66	0.57	0.56	0.55	0.57	0.56
Davies-Bouldin Score	0.48	0.53	0.55	0.55	0.47	0.46
Calinski-Harabasz Score Score	505.43	561.82	705.15	787.05	900.48	1187.55
V Measure Score	0.43	0.43	0.37	0.41	0.40	0.38
ARI Score	0.37	0.37	0.29	0.31	0.29	0.22
AMI Score	0.42	0.42	0.36	0.40	0.39	0.36

***Table 15:** Results from K-Means for number of clusters (Wine dataset)*

For the Wine dataset, it can be observed that the optimal number of clusters is either 2 or 3. It can be noted that the clusters are equally homogenously and completely labelled when *k* is set to 2 and 3 from the V measure, ARI, and AMI scores. However, the interesting observations here would be how silhouette score is favored to a lower number of clusters while the DB and CH score favors higher number of clusters.

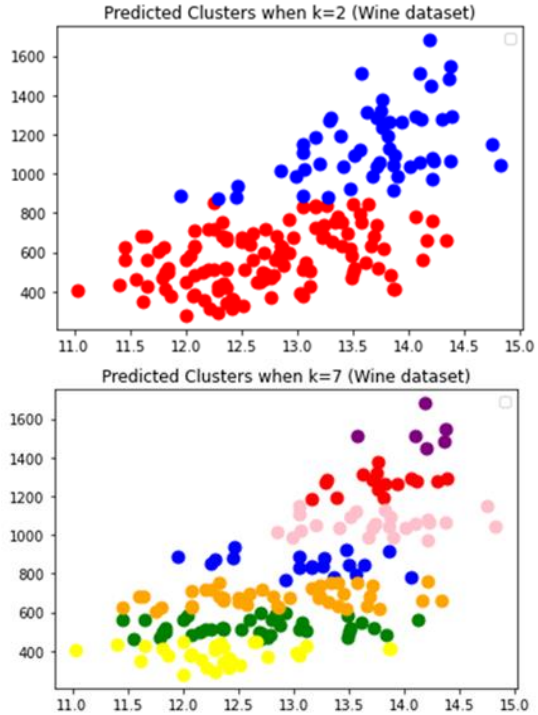

***Figure 10**: Predicted clusters when k=2 (top) and k=7 (bottom) for Wine dataset*

When *k* is initialized to 2, the clusters are more spread apart from each other and distinguished, which may explain the favoring of small number of clusters for the silhouette score. On the contrary, when the number of clusters is set

to 7, the clusters are more spread apart and the average similarity of each cluster with a cluster most similar to it is minimized, leading to a better DB score. As the clusters are more spread apart from each other, the similarity of an object to its own cluster will be higher than that of another cluster, which results in a higher CH score for a higher number of clusters for the Wine dataset.

Algorithm used

For K-means, two algorithms can be employed – Lloyd and Elkan. By default, the chosen algorithm is Lloyd, but for the purpose of this analysis, we decided to employ both algorithms (with *k* initialized to 3) to compare which algorithm is better at clustering the datasets.

The Lloyd's algorithm is based on the observation that, while jointly optimizing clusters and assignment is difficult, optimizing one given the other is easy. It alternates two steps – Quantization, where each point is reassigned to the center closer to it and requires finding for each point the closest among other points, and Centre estimation, where each center is updated to minimize its average distances to the points assigned to it. Elkan's algorithm, on the other hand, is a variation of the Lloyd's algorithm. The Elkan's algorithm uses the triangular inequality to avoid many distance calculations when assigning points to clusters.

Algortihm	Elkan	Lloyd
Accuracy	0.89	0.89
Silhouette Score	0.55	0.55
Davies-Bouldin Score	0.66	0.66
Calinski-Harabasz Score	560.39	560.39
V Measure Score	0.75	0.75
ARI Score	0.73	0.73
AMI Score	0.75	0.75

Table 16: *Results from K-Means for Algorithm used (Iris dataset)*

Algortihm	Elkan	Lloyd
Accuracy	0.70	0.70
Silhouette Score	0.57	0.57
Davies-Bouldin Score	0.53	0.53
Calinski-Harabasz Score	561.81	561.81
V Measure Score	0.43	0.43
ARI Score	0.37	0.37
AMI Score	0.42	0.42

Table 17: *Results from K-Means for Algorithm used (Wine dataset)*

Based on our observations, it can be seen that there is no significant difference to the clustering when employing the two algorithms based on the performance metrics described. However, the notable difference in the algorithms would be the runtime as the Elkan algorithm generally runs faster than the Lloyd algorithm, but the algorithm is unsuitable for large number of clusters as it uses storage proportional to the number of clusters by data points.

Initialization of centroid

In the generic K-means algorithm, the initialization of centroids is done randomly, which is a factor that affects the overall performance of the clustering method itself as the method is sensitive to the initialization of centroids. As such, we employed the K-means++ centroid initialization to determine if the method will perform better with a different method of centroid initialization. K-means++ initializes centroids that are far away from each other by sequentially initializing the centroids. It chooses the data point with the highest D(x), where D(x) is the shortest distance from a data point to the closest centroid already chosen, as the new centroid. This process will repeat until all *k* centroids have been chosen.

Initialization	Random	K-Means++
Accuracy	0.89	0.89
Silhouette Score	0.55	0.55
Davies-Bouldin Score	0.66	0.66
Calinski-Harabasz Score	560.39	560.39

V Measure Score	0.75	0.75
ARI Score	0.73	0.73
AMI Score	0.75	0.75

Table 18: *Results from K-Means for cluster initialization (Iris dataset)*

Initialization	**Random**	**K-Means++**
Accuracy	0.70	0.70
Silhouette Score	0.57	0.57
Davies-Bouldin Score	0.53	0.53
Calinski-Harabasz Score	561.81	561.81
V Measure Score	0.43	0.43
ARI Score	0.37	0.37
AMI Score	0.42	0.42

Table 19: *Results from K-Means for cluster initialization (Wine dataset)*

From our observations, there is no significant difference in both initializations based on the metrics employed.

4.3.2 Results of K-means clustering

Based on our experiments, it can be noted that there is no significant difference in the initialization of centroids and algorithm used, and the optimal number of centroids for the Iris dataset is 3 and the optimal number of centroids for the Wine dataset is either 2 or 3. For the optimal model employed to cluster both datasets, we will be using the Elkan's algorithm, random centroid initialization and *k* set to 3.

Iris dataset

Number of Clusters	**3**
Algorithm	**Elkan**
Initialization	**Random**
Accuracy	0.89
Silhouette Score	0.55
Davies-Bouldin Score	0.67
Calinski-Harabasz Score	560.39
V Measure Score	0.75
ARI Score	0.73
AMI Score	0.75

Table 20: *Results from K-Means for Iris dataset*

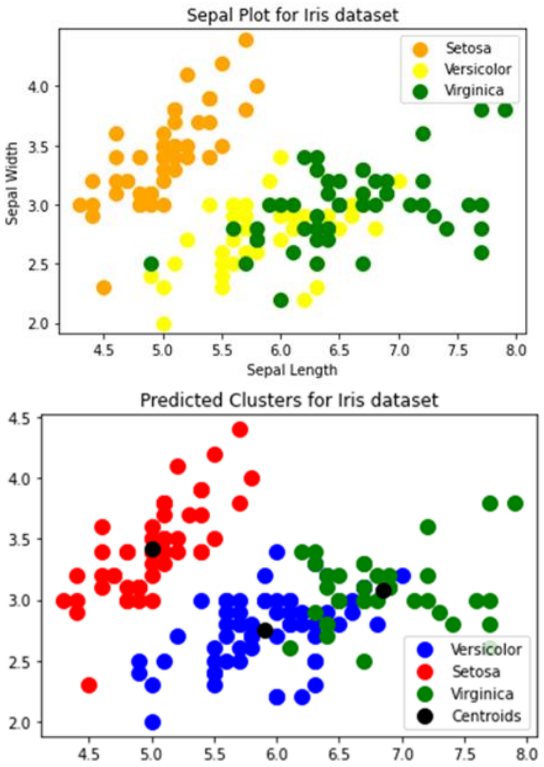

Figure 11*: Scatter plot for Iris dataset before clustering (top), after clustering (bottom)*

As seen from the Table 20, K-means performed relatively well for the clustering of Iris dataset, with an accuracy of 0.89 against its ground-truth values.

Wine dataset

Number of Clusters	**3**
Algorithm	**Elkan**
Initialization	**Random**
Accuracy	0.70
Silhouette Score	0.57
Davies-Bouldin Score	0.53
Calinski-Harabasz Score	561.81
V Measure Score	0.43
ARI Score	0.37
AMI Score	0.42

Table 21: *Results from K-Means for Wine dataset*

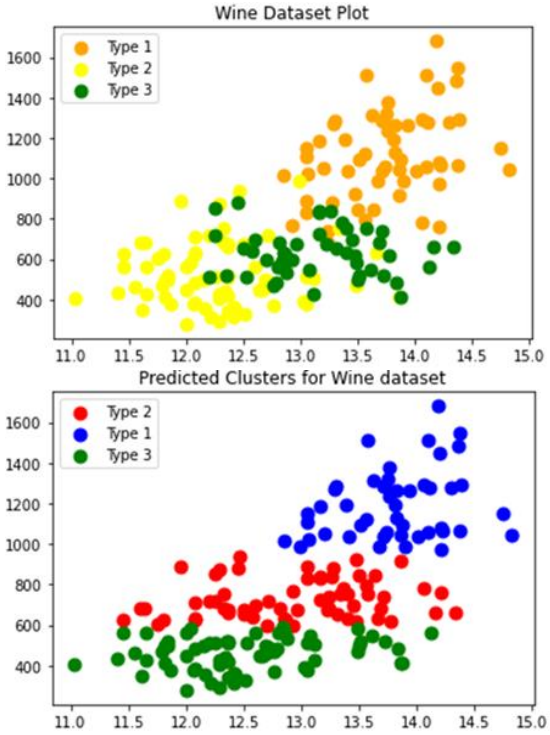

***Figure 12**: Scatter plot for Wine dataset before clustering (top), after clustering (bottom)*

As seen from Table 21, K-means also performed relatively well for the clustering of the Wine dataset, with an accuracy of 0.70 against its ground-truth values.

4.4 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an unsupervised learning technique used to group data based on similar characteristics. DBSCAN requires two parameters: epsilon (ε) and minPoints (*n*). Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

To create clusters, DBSCAN randomly picks a point from the dataset and assigns it to a cluster. Then it counts how many points are located within the epsilon distance from the selected point. If the number is greater than or equal to minPoints, then it is considered a core point, after which all neighbours are put in the same cluster. If some member of this cluster has at least *n* neighbours, it will expand the same cluster by putting the neighbours into the same cluster. It will continue expanding the first cluster until there are no more neighbours to put in. Then, it will pick another point from the dataset which does not belong to any cluster and put it in the second cluster. The above process until all data points in the dataset belong to some cluster or are marked as outliers (noise). Data points are then classified into 3 categories: Core point, Border Point, and Noise Point, based on the previous parameters, ε and *n*. A core point is a data point that has at least minPoints within epsilon distance. A border point is a data point that has at least one core point within the epsilon distance and is lower than minPoints within the epsilon distance from it, and a noise point is a data point that has no core points within the epsilon distance.

DBSCAN is very sensitive to the values of epsilon and minPoints. Therefore, it is important to understand how to choose the values of epsilon and minPoints.

MinPoints

There is no automatic way to determine the minPoints, but there are general rules to follow in order to choose the minPoints value. MinPoints should be greater than or equal to the number of dimensions, (*D*) + 1. The larger the data set or the noisier the data set, the larger the value of minPoints. Therefore, minPoints can be generally defined as 2**D*, where *D* is the number of dimensions, then adjusted based on the dataset.

Epsilon

The optimal value of epsilon is the elbow value of the nearest neighbour graph [22]. However, if the value of epsilon chosen is too small, a large part of the data will not be clustered. On the other hand, if the value of epsilon chosen is too high, clusters will merge and the majority of objects will be in the same cluster. In general, small values of epsilon are preferable.

Strengths:

1. DBSCAN algorithm is robust to outliers.
2. DBSCAN does not require a pre-determined set number of clusters.
3. DBSCAN performs well with arbitrary-shaped clusters.

Weaknesses:

1. DBSCAN cannot cluster data sets well with large differences in densities.
2. If the data and scale are not well understood, choosing a meaningful epsilon can be difficult.

4.4.1 Experiments with Various Parameters

Epsilon and minPoints

The values of epsilon and minPoints can be pre-determined from the method mentioned above, but we decided to vary the values of epsilon and minPoints to find the optimal parameter settings based on the performance metrics described.

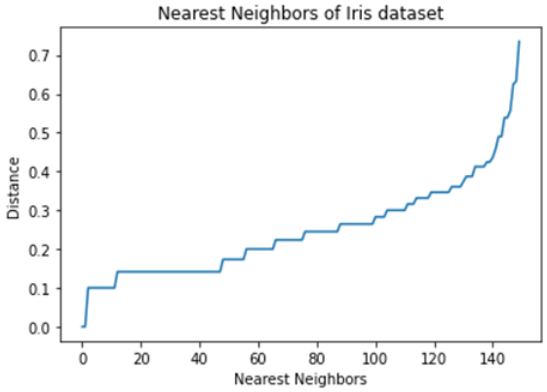

***Figure 13**: Nearest neighbour graph of Iris dataset*

From the nearest neighbour method, we determined the optimal value of epsilon to be around 0.5 for the Iris dataset, hence we ran the DBSCAN algorithm for epsilon values in a range of 0.4 to 0.8, and minPoints values in range of 2 to 25 for the Iris dataset.

Epsilon	**0.40**	**0.40**	**0.60**	**0.60**	**0.80**
MinPoints	**3**	**4**	**8**	**24**	**25**
Accuracy	0.57	0.79	0.63	0.32	0.67
Silhouette Score	0.33	0.33	0.54	0.66	0.51
Davies-Bouldin Score	2.92	2.82	11.61	0.40	0.45
Calinski-Harabasz Score Score	123.19	122.48	219.20	395.54	277.99
V Measure Score	0.71	0.70	0.62	0.66	0.72
ARI Score	0.71	0.68	0.53	0.52	0.56
AMI Score	0.70	0.69	0.62	0.66	0.71

***Table 22:** Results from DBSCAN for epsilon and minPoints (Iris dataset)*

From the results, the optimal parameters settings chosen for the Iris dataset is an epsilon value of 0.6 and minPoints value of 8 as this setting obtained a good average for all of the performance metric scores.

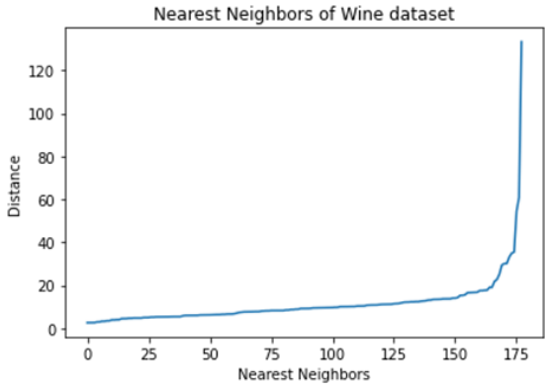

***Figure 14**: Nearest neighbour graph of Wine dataset*

From the nearest neighbour method, we determined the optimal value of epsilon to be around 40 for the Wine dataset, hence we ran the DBSCAN algorithm for epsilon values in a range of 40 to 43, and minPoints values in range of 2 to 15.

Epsilon	**40**	**41**	**43**
MinPoints	**5**	**14**	**15**
Accuracy	0.53	0.48	0.35
Silhouette Score	0.55	0.39	0.59
Davies-Bouldin Score	0.75	29.08	0.58
Calinski-Harabasz Score Score	275.74	163.91	329.66
V Measure Score	0.38	0.41	0.43
ARI Score	0.24	0.35	0.43
AMI Score	0.36	0.40	0.43

***Table 23:** Results from DBSCAN for epsilon and minPoints (Wine dataset)*

From the results, the optimal parameters settings chosen for the Wine dataset is an epsilon value of 43 and minPoints value of 15 as this setting obtained the majority of the best performance metric scores.

3.4.2 Results of DBSCAN clustering

Iris dataset

Epsilon	0.6
MinPoints	**8**
Accuracy	0.63
Silhouette Score	0.54
Davies-Bouldin Score	11.61
Calinski-Harabasz Score	219.20
V Measure Score	0.62
ARI Score	0.53
AMI Score	0.62

Table 24: *Results from DBSCAN for Iris dataset*

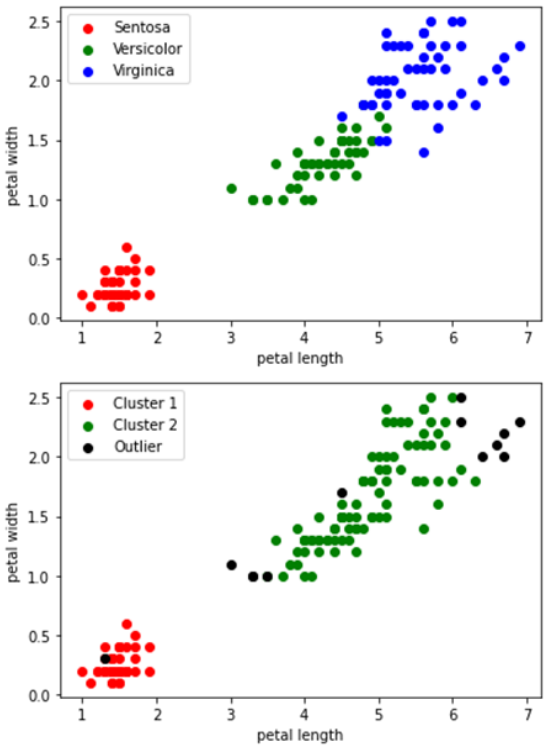

Figure 15*: Scatter plot for Iris dataset before clustering (top), after clustering (bottom)*

After running DBSCAN with the optimal parameters, the results are shown in Table 24 and Figure 15. It can be observed that the number of clusters reduced from 3 to 2 after clustering by DBSCAN. This exemplifies the weakness of DBSCAN – the number of clusters cannot be controlled. When clusters with similar values is lined near each other, DBSCAN is unable to separate the clusters, but instead merges them into one big cluster.

Wine dataset

Epsilon	43
MinPoints	**15**
Accuracy	0.35
Silhouette Score	0.59
Davies-Bouldin Score	0.58
Calinski-Harabasz Score	329.66
V Measure Score	0.43
ARI Score	0.43
AMI Score	0.43

Table 25: *Results from DBSCAN for Wine dataset*

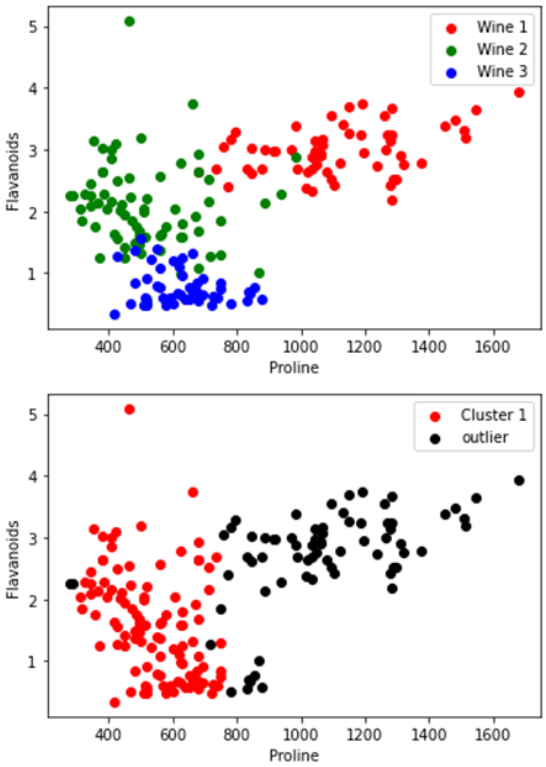

Figure 16*: Scatter plot for Wine dataset before clustering (top), after clustering (bottom)*

The results from Figure 16 shows how the DBSCAN algorithm performed on the Wine dataset. It can be observed that there is only one large cluster, with the rest of the datapoints being set as outliers. This further exemplifies another weakness of DBSCAN – its inability to perform well on multi-dimensional data.

5 Experimental Results

To compare how well each algorithm fared against each other, we decided to use 3 performance metrics as the standard of evaluation. First, we compared accuracy of the algorithm, which is able to tell us how each algorithm fared in terms of the labelling against

its ground-truth value. Secondly, we chose to use V Measure score to compare how homogenously and completely the algorithms clustered the data points in both datasets. Lastly, we used the CH score to determine how densely and well-separated the algorithms clustered the data points in the dataset.

Iris dataset

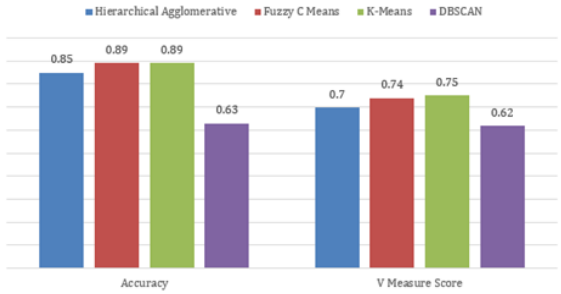

***Figure 17**: Bar plot for accuracy and V Measure score (Iris dataset)*

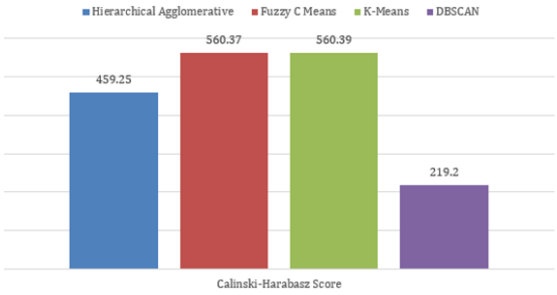

***Figure 18**: Bar plot for CH Score (Iris dataset)*

K-means and FCM produced the best outcomes for each of the performance metrics used. However, it can be noted that K-Means edged out FCM just slightly for both V Measure scores and CH scores by 0.01 and 0.02 respectively, but both methods produced the same accuracy. This might indicate that for the Iris dataset, partitional clustering methods such as K-means and FCM, may be better suited instead of other clustering methods such as hierarchical agglomerative and DBSCAN. That being said, agglomerative clustering produced decent results as well, being only slightly lower than FCM with a 0.85 accuracy, a 0.70 V measure score and a 459.25 C-H score.

Wine dataset

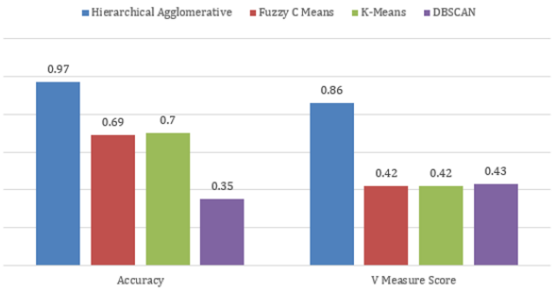

***Figure 19**: Bar plot for accuracy and V Measure score (Wine dataset)*

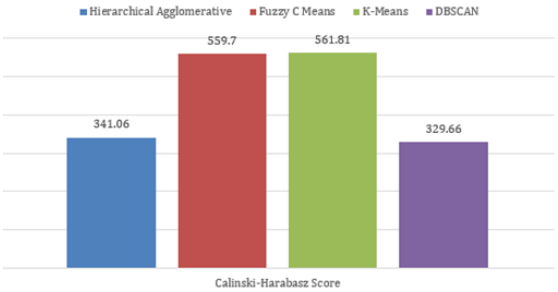

***Figure 20**: Bar plot for CH Score (Wine dataset)*

For the Wine dataset, hierarchical agglomerative clustering method produced the best results for accuracy and V Measure score, as noted in Figure 18. Other models have seemingly dropped in performance across all metrics, especially accuracy and V measure, when compared to their performance in clustering the Iris dataset. Hierarchical agglomerative clustering, on the other hand, has an improved performance when using the Wine dataset.

While the high accuracy and V measure scores would suggest that the clusters are rightly, homogenously, and completely clustered using hierarchical agglomerative clustering, the lowered CH scores suggest that the clusters may not be as well separated as clusters from other models. This suggests that even though hierarchical agglomerative clustering has produced the most accurate model, the clusters the model initialized is not as dense as K-means or Fuzzy (both of which performed better in CH score). A denser cluster is usually indicative of a better cluster as the cluster would have more data points that are closer to each other. The results, however, might suggest a different outcome. The denser cluster has achieved a much lower accuracy and V measure score. This could be due to the presence of outliers from each category being similar to data points from other categories, thus leading to the model cluster the data points in a different manner. From this, we can conclude that hierarchical

agglomerative clustering may be able to handle outliers better than other clustering methods.

6 Conclusion

Through our research and findings, we found out that different models have different outcomes when using different datasets. This suggests that for clustering algorithms, there is no 'one size fits all' and that for different tasks and datasets, different clustering methods can prove to be more effective in handling a specific scenario.
Clustering proves to be a difficulty for algorithms as there are a myriad of factors that affect its performance. In the best-case scenario, clustering should be unsupervised, void of human intervention, yet many factors require it. From choosing the individual parameters of each method to choosing the method itself, there is a need for user interference. More research is definitely required to identify which scenario requires which model and ways to allow the algorithm to select its own parameters.

7 References

[1] S. H. Shah, M. J. Iqbal, M. Bakhsh and A. Iqbal, "Analysis of Different Clustering Algorithms for Accurate Knowledge Extraction from Popular DataSets," *Information Sciences Letters*, vol. 1, pp. 21-31, 2020.

[2] P. Nerurkar, A. Shirke, M. Chandane and S. Bhirud, "Empirical Analysis of Data Clustering Algorithms," *Procedia Computer Science*, vol. 125, pp. 770-779, 2018.

[3] R. Gelbard, O. Goldman and I. Spiegler, "Investigating diversity of clustering methods: An empirical comparison," *Data & Knowledge Engineering*, vol. 63, no. 1, pp. 155-166, 2007.

[4] A. Gosain and S. Dahiya, "Performance Analysis of Various Fuzzy Clustering Algorithms: A Review," *Procedia Computer Science*, vol. 79, pp. 100-111, 2016.

[5] C. Doring, M.-J. Lesot and R. Kruse, "Data Analysis with Fuzzy Clustering Methods," *Computational Statistics & Data Analysis*, vol. 51, no. 1, pp. 192-214, 2006.

[6] C. C. B. Jr., "Hierarchical Cluster Analysis," *Psychological Reports*, vol. 18, no. 3, pp. 851-854, 1966.

[7] E. Burghardt, D. Sewell and J. Cavanaugh, "Agglomerative and divisive hierarchical Bayesian clustering," *Computational Statistics & Data Analysis*, vol. 176, 2022.

[8] D. Dua and C. Graff, "Iris Dataset," University of California, School of Information and Computer Science , Irvine, CA, 2019.

[9] D. Dua and C. Graff, "Wine Data Set," University of California, School of Information and Computer Science, Irvine, CA, 2019.

[10] "Selecting the number of clusters with silhouette analysis on KMeans clustering," sckit-learn, [Online]. Available: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html.

[11] "sklearn.metrics.davies_bouldin_score," sckit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html.

[12] "Calinski-Halibasz Index - Cluster Validity indices," geeksforgeeks, 25 April 2022. [Online]. Available: https://www.geeksforgeeks.org/calinski-harabasz-index-cluster-validity-indices-set-3/. [Accessed 15 October 2022].

[13] "sklearn.metrics.v_measure_score," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html. [Accessed 15 October 2022].

[14] "sklearn.metrics.homogeneity_score," scikit-learn, [Online]. Available: https://scikit-

learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html. [Accessed 15 October 2022].

[15] "sklearn.metrics.completeness_score," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html. [Accessed 15 October 2022].

[16] "sklearn.metrics.adjusted_rand_score," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html. [Accessed 15 October 2022].

[17] "sklearn.metrics.adjusted_mutual_info_score," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html. [Accessed 15 October 2022].

[18] "Accuracy (error rate)," DeepAI, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate. [Accessed 15 October 2022].

[19] P. Sharma, "A Beginner's Guide to Hierarchical Clustering and how to perform it in Python," Analytics Vidhya, 27 May 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/. [Accessed 15 October 2022].

[20] P. Wilkinson, "Introduction to Hierarchical clustering (part 1 — theory, linkage and affinity)," Towards Datascience, 5 March 2021. [Online]. Available: https://towardsdatascience.com/introduction-to-hierarchical-clustering-part-1-theory-linkage-and-affinity-e3b6a4817702. [Accessed 15 October 2022].

[21] S. Miyamoto, R. Abe, Y. Endo and J.-i. Takeshita, "Ward method of hierarchical clustering for non-Euclidean similarity measures," *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR),* pp. 60-63, 2015.

[22] N. Rahmah and I. S. Sitanggang, "Determination of Optimsal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra," *IOP Conference Series: Earth and Environmental Science,* vol. 31, pp. 1-5, 2016.