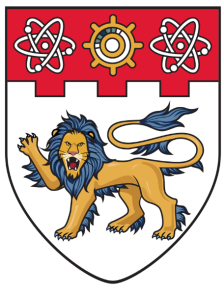

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# CZ4032 - Data Analytics and Mining

## Project 2 (Group 27)

### Team Members:


Arya Shashwat (U1822436J)

Singh Kirath (U1822329J)

Chatterjee Pramurta (U1822597G)

Hasan Mohammad Yusuf (U1822478E)

# Data Clustering Algorithms - A Comparative Analysis

Singh Kirath
U1822329J
kirath001@e.ntu.edu.sg

Chatterjee Pramurta
U1822597G
pramurta001@e.ntu.edu.sg

Hasan Mohammad Yusuf
U1822478E
mohammad059@e.ntu.edu.sg

Arya Shashwat
U1822436J
shashwat002@e.ntu.edu.sg

## 1. Abstract

With the boom of big data being used, data mining is becoming more important by the day. One particular data mining technique used to group data points into natural groups based on their attributes is called clustering. There are several clustering algorithms available, such as K-Means, K-Means++ and DBscan. This paper attempts to use these algorithms on 2 separate datasets, and calculate a performance metric from the clusters obtained. The performance metric would be the accuracy derived from the clusters obtained and the predefined class labels for each dataset.

## 2. Introduction

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups using unsupervised machine learning. It is a main task of exploratory data analysis, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

There are several clustering algorithms available to choose from, such as connectivity models (hierarchical clustering), centroid models (K-Means, K-Means++ clustering) and density models (DBSCAN, OPTICS). All these models have their own advantages and drawbacks, and several studies have been conducted to measure their efficiency. But whenever comparing clustering algorithms, there is a lack of a specific accuracy metric to evaluate the performance of an algorithm on a particular dataset, partly due to clustering being an unsupervised technique. This paper tries to bridge this gap by proposing a method to calculate the accuracy for 2 such clustering algorithms - DBSCAN and K-Means, and then comparing their results against each other. This would be done by comparing the clusters obtained against the predefined class labels, essentially making clustering partly supervised. Then using this accuracy metric, we propose a simple grid search of all hyper-parameters to find the optimal values for each.

## 3. Literature review

When comparing clustering algorithms, previous research compares the performance of clustering algorithms based on metrics such as time complexity and dataset size [1]. These measures are then used to determine the merits and demerits associated with each algorithm.

A measure of numerical accuracy is excluded in the final comparison.

Another study tries to compare model performance using synthetic datasets in order to account for the many possible variations of data [2]. This approach is effective, however it relies on choosing the hyper-parameters randomly. There is no control over iteratively selecting the best parameters, and one has to rely on the randomized approach selecting the optimal parameters.

A study conducted by Stephen Becker attempts to use PCA for dataset compression and then applying K-Means on the compressed dataset to generate the error in the center estimators at a given step [3]. Their research paper shows that the K-Means algorithm provides a real benefit when applied to standard test data sets, and provides certain benefits over other sampling approaches. They however performed their tests on only one clustering approach, learning some scope for further analysis on more algorithms like DBscan.

Another study authored by W. Chen et. al. aims to develop a modified density-based clustering algorithm, named T-DBSCAN, by considering the time sequential characteristics of points along a trajectory and comparing its performance against K-Means clustering [4]. Their paper discusses the performance and drawbacks of their new proposed algorithm when implemented to find the trajectory on a geographical map. Their study yet lacks a numerical comparison between the 2 clustering approaches and mostly discusses their benefits and shortcomings.

**Our Approach:** Most of these studies only compare the performance using non-numeric metrics such as time complexity and dataset size or scrutinize the performance of a single clustering algorithm like K-Means. There is still a lack of a fixed numeric accuracy metric to compare the performance of multiple clustering algorithms against a dataset. We predict the clustering model accuracy based on class labels, and define a numeric accuracy for model comparison across datasets.

## 4. Clustering methods

For the comparison of clustering algorithms we have 2 popular clustering algorithms - DBscan and K-Means. Both algorithms have their own advantages and disadvantages and their performance depends on the density and distribution of the dataset chosen.

### 4.1 DBscan (Density-Based Spatial Clustering of Applications with Noise)

As proposed by Martin Ester et al. in 1996, DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density. It should be used to find associations and structures in data that are hard to find manually but that can be relevant and useful to find patterns and predict trends.

It groups ‘densely grouped’ data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. An important feature of DBSCAN clustering is that it is robust to outliers. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

DBscan creates a circle of epsilon radius around every data point and classifies them into Core point, Border point, and Noise [5]. A data point is a Core point if the circle around it contains at least “min_samples” number of points. If the number of points is less than minPoints, then it is classified as Border Point, and if there are no other data points around any data point within

epsilon radius, then it is treated as Noise. This phenomenon can be seen in the figure below.

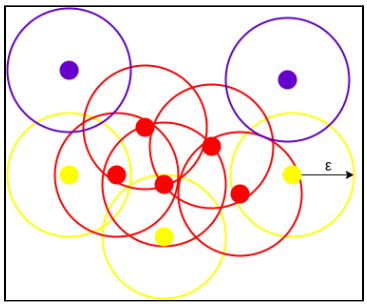


Fig. 1 - epsilon radius per data point

DBscan requires 2 parameters be se set at initialization-

1. **Epsilon (eps)**
   It is the radius of the circle to be created around each data point to check the density

2. **Min_samples**
   It is the minimum number of data points required inside that circle for that data point to be classified as a “core” point.

**Advantages of DBscan clustering**

1. **Auto cluster numbers:** DBSCAN is an alternative to k-means, which requires one to specify the number of clusters in the data a priori. The number of clusters are decided dynamically during the execution.

2. **Arbitrary cluster shape:** DBSCAN can find arbitrarily-shaped clusters. It can even find a cluster completely surrounded by a different cluster (like in the figure below).

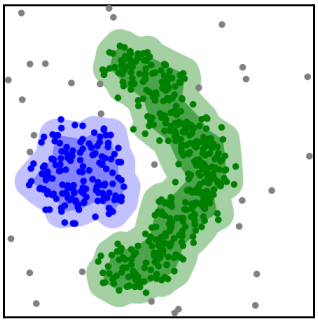

Fig. 2 - cluster partly surrounded by another cluster

3. **Noisy data:** DBSCAN has a notion of noise, and is robust to outliers as long as they do not concentrate in the inter-cluster boundaries.

**Disadvantages of DBscan clustering**

1. **Variable density datasets:** DBSCAN cannot cluster data sets well with large differences in densities, since the min_samples combination cannot then be chosen appropriately for all clusters.

2. **Neak-type dataset:** Fails in case of neck type of dataset (like in the figure below). This is because the points at the boundary of both clusters satisfy the min_sample points from either cluster. This behaviour is extended to points in each cluster, leading to only a single cluster being recognized by DBscan.

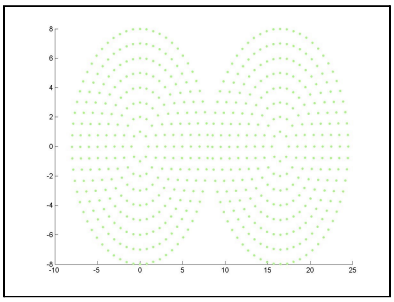

Fig. 3 - neck type dataset

3. **High data dimension:** DBscan does not work well in case of high dimensional data as selecting a meaningful epsilon value is difficult as the euclidean distance depends on multiple attributes.

## 4.2 K-Means

Clustering has a property that the points within a cluster should be comparable to one another. As a result, our goal is to reduce the distance between points inside a cluster.

K-Means is a centroid-based or distance-based technique in which the distances between points are calculated to allocate a point to a cluster. Each cluster in K-Means is paired with a centroid. The K-Means algorithm's main goal is to reduce the sum of distances between points and their corresponding cluster centroid.

The following are the **steps of utilizing K-Means to construct clusters**:

1. Choose the number of clusters (the 'K' in K-Means).

2. Each cluster's centroid is chosen at random. Assume we want two clusters, therefore k is equal to two. The centroid is then chosen at random.

3. We assign each point to the closest cluster centroid once the centroids have been initialized.

4. After assigning all of the points to one of the two clusters, we compute the centroids of the newly created clusters.

5. Steps 3 and 4 should be repeated until the stopping criteria are reached.

**Stopping criteria** is required to end the iterations taking place during the process of training. The three most commonly used stopping criterias are as follows:

1. If the centroids of newly generated clusters do not change, we can end the procedure. If we get the same centroids for all the clusters after numerous rounds, we can conclude that the algorithm is not learning any new pattern and that we should terminate training.

2. Another indication that we should halt the training process is if the points remain in the same cluster after several rounds of the algorithm.

3. Finally, after the maximum number of iterations has been reached, the training can be terminated. Assume that the number of iterations is set to 100. Before stopping, the process will repeat for 100 iterations.

**Challenges with K-Means algorithm**

1. The size of clusters formed are different. Let's say we have the below points:

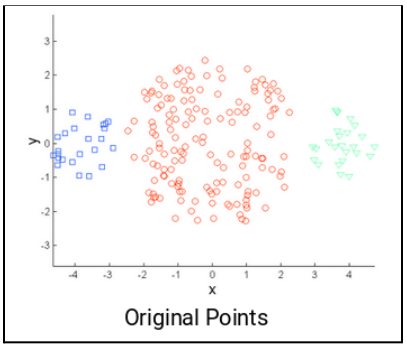


Fig. 4 - Original data points

In comparison to the centre cluster, the left and rightmost clusters are smaller. Now, if we use k-means clustering to group these points together, we get something like this:

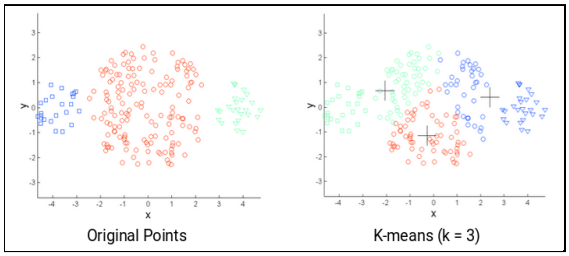


Fig. 5 - Comparison between original data points and clusters

2. The densities of the original points are different. Let's say these are the original points:

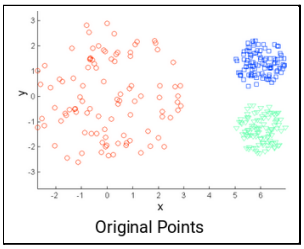


Fig. 6 - Another sample of Original data points

The red cluster's points are spaced out, whereas the remaining clusters' points are tightly packed together. When we apply k-means to these points, we get the following clusters:

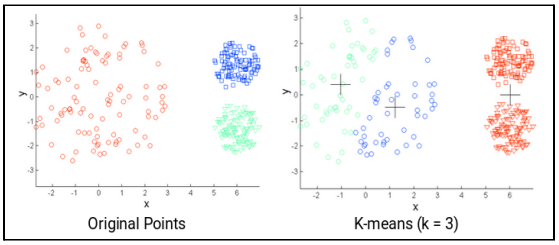


Fig. 7 - Comparison between original data points and clusters

The compact points have been assigned to a single cluster, as can be seen. Points that are dispersed yet belong to the same cluster have been assigned to distinct clusters.

3. Random initialization of centroids can be problematic because we might get different clusters every time.

## Solutions to overcome the challenges

1. **K-Means++ to Choose Initial Cluster Centroids for K-Means Clustering**

   K-Means can produce arbitrarily terrible clusters in some instances if cluster initialization is not done properly. This is when K-Means++ comes in handy. Before using the typical k-means clustering algorithm, it outlines a procedure for initializing the cluster centers.

   We use the K-Means++ approach to improve the step where we choose the cluster centroid at random. Using the K-Means++ initialization, we are more likely to obtain a solution that is competitive with the ideal K-Means solution.

   The steps to initialize the centroids using K-Means++ are:

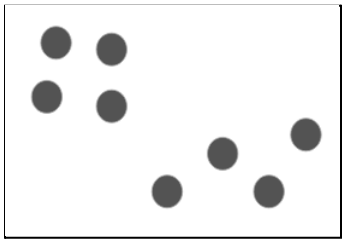

Fig. 8 - Data Points

1. Randomly pick a data point as a cluster centroid

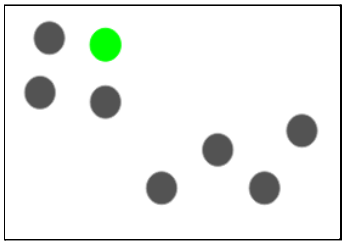

Fig. 9 - Pick centroid

2. Let's say the initial centroid is the green point. We'll now calculate the distance between each data point and this centroid (D(x))

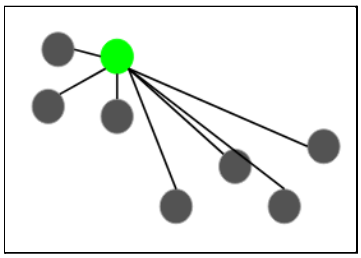

Fig. 10 - Calculate Distance

3. The next centroid will be the one that is the furthest away from the current centroid in terms of squared distance (D(x)^2)

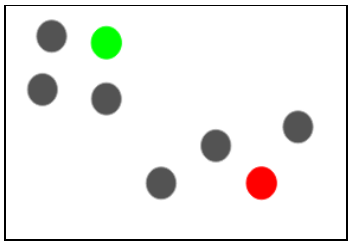

Fig. 11 - Find next centroid

4. The red point will be chosen as the next centroid in this situation. To determine the last centroid, we will measure each point's distance from its nearest centroid, and the location with the greatest squared distance will be chosen as the next centroid

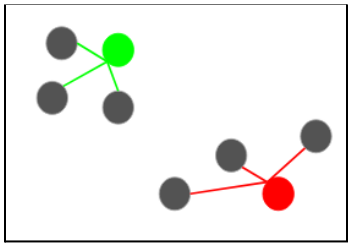

Fig. 12 - Calculate distances

5. We will select the last centroid.

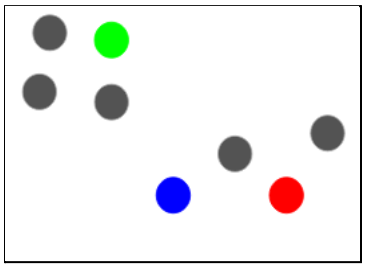

Fig. 13 - Select last centroid

After initializing the centroids, we can continue with the K-Means algorithm. When the centroids are initialized with K-Means++, the clusters tend to improve. Although it is more computationally expensive than random initialization, subsequent K-Means frequently converge faster.

**2. Elbow method to find optimal number of clusters**

The x-axis will reflect the number of clusters, and the y-axis will be an evaluation metric in the Elbow Curve. As an evaluation metric, we'll use Inertia.

Steps to make an Elbow Curve:

For example, let's start with a tiny cluster value of 2. Train the model with two clusters, determine its inertia, and lastly plot it in the graph above. Let's imagine we get an inertia value of 1000.

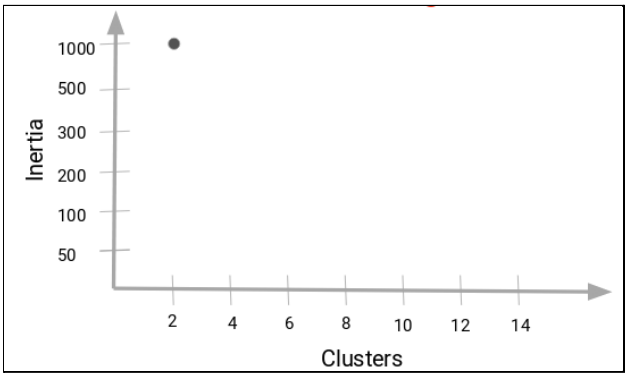


Fig. 14 - Plot inertia for k=2

We'll now increase the number of clusters, retrain the model, and plot the inertia value. The plot is as follows:

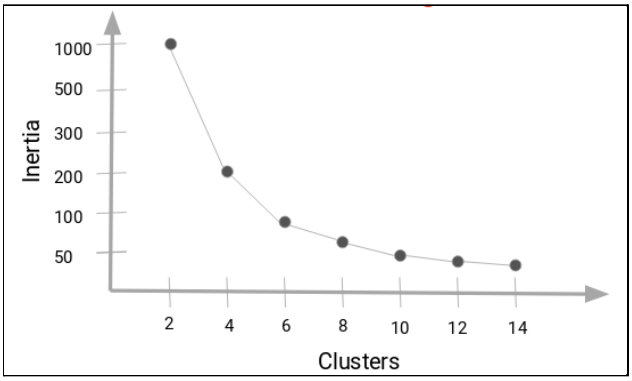


Fig. 15 - Plot inertia values for every K

The inertia value dropped drastically when the cluster value was increased from 2 to 4. As we increase the number of clusters, this decrease in inertia value decreases and eventually becomes constant. As a result, the cluster value at which the drop in inertia value is constant can be chosen as the correct cluster value for our data.

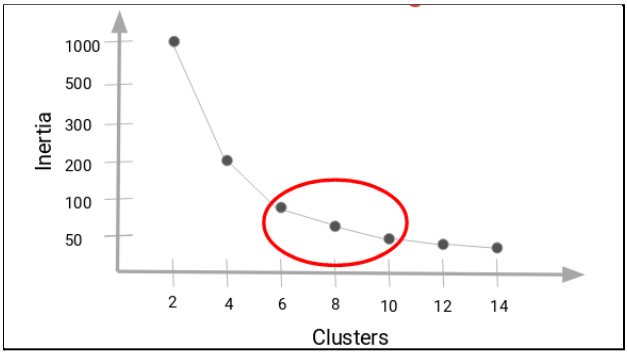


Fig. 16 - Find optimal K

Any number of clusters between 6 and 10 can be chosen. We can have as many as seven, eight, or even nine clusters. When deciding on the number of clusters, we must also consider the cost of computing. The cost of computation will rise as the number of clusters increases. As a result, if computational resources are limited, we should select fewer clusters.

**Advantages of K-Means clustering**

1. **High Performance:** The K-Means approach has a linear time complexity and may easily be applied to huge datasets. As an unsupervised clustering algorithm, K-Means provides many insights and benefits when dealing with unlabeled huge data.

2. **Easy to use:** K-Means is also simple to operate. In the Scikit-Learn implementation, default settings can be used to initialize it. Parameters such as the number of clusters (8 by default), the maximum iterations (300 by default), and the initial centroid initialization (10 by default) can all be readily changed later to meet the job goals.

3. **Result Interpretation:** K-Means produces clusters that are simple to understand and even visualize. Because of its simplicity, it can be quite beneficial in situations where you require a rapid overview of the data parts.

   In addition, the inertia values generated by the K-Means algorithm can be interpreted. Sum of squared means of K-Means inertia for each point to its cluster center (centroid). Higher inertia values can be useful in debating the inner workings of a cluster number or algorithm, such as initialization or maximum iteration.

**Disadvantages of K-Means clustering**

1. **Results Repeatability:** Because of the random centroid initialization, one of the inconsistencies of the K-Means technique is that the outcomes will differ.

   Unless you pick the centroids at set positions, which is uncommon. After iterations, K-Means can produce a variety of clusters.

   In addition, if you add new data or rearrange an old dataset, K-Means is likely to provide different findings. Because of this, K-Means is a less-than-stable machine learning algorithm.

2. **Spherical Clustering Only:** Spherical clusters are generated using K-Means. As a result, K-Means will not be able to cluster overlapping clusters or arbitrary forms.

3. **Clusters everything:** Another feature of K-Means is that every data sample is included in the clusters it generates. This means that using the K-Means approach, which builds spherical clusters that encompass the entire dataset, you won't be able to remove outliers or specific sample groups.

## 5. Datasets used

We have used 2 datasets from the UCI machine learning for our clustering experiments. Both datasets have a class label for each data point, which can be easily used to test the clusters generated by our classifier against the actual class clusters in the dataset. The datasets chosen are

### 5.1 Seeds dataset

This dataset classifies kernels belonging to 3 different varieties of wheat - ‘Kama’, ‘Rosa’ and ‘Canadian’. Each row contains 7 real-values continuous attributes related to the geometric size of the wheat kernel. There are a total of 210 data points, 70 for each class label for fair distribution.

The diagram below showcases the cluster distribution of the 3 types of seed kernels.

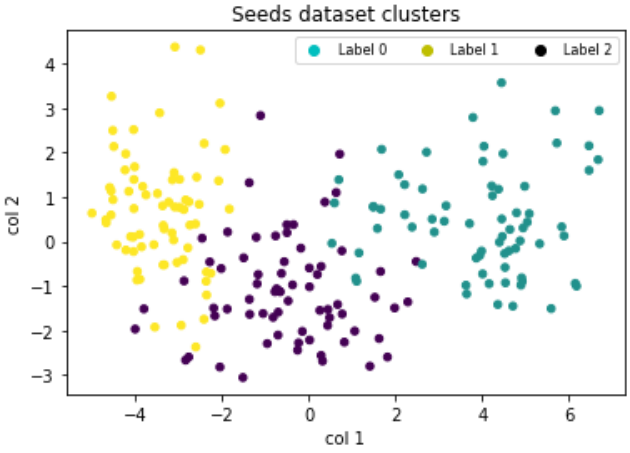


Fig. 17 - clusters in the original “seeds” dataset

Since the data points are far apart and within well defined clusters, this dataset would give a base insight into each clustering algorithm’s results.

### 5.2 Forest dataset

This dataset contains data from a remote sensing study which mapped different forest types. The output (forest type map) can be used to identify and/or quantify the ecosystem services (e.g. carbon storage, erosion protection) provided by the forest. Each data point in the dataset contains 27 real-valued continuous attributes about a forest’s spectral characteristics at visible-to-near infrared wavelengths and one of 4 class labels - 's' ('Sugi' forest), 'h' ('Hinoki' forest), 'd' ('Mixed deciduous' forest), 'o' ('Other' non-forest land). The dataset contains a total of 325 rows.

The diagram below showcases the cluster distribution of the 4 types of forests.

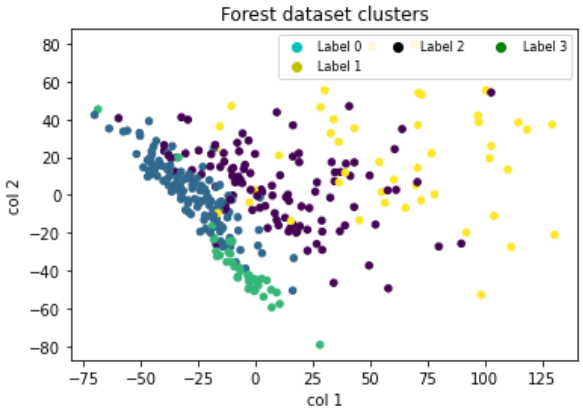


Fig. 18 - clusters in the original “forest” dataset

This dataset contains clusters with varying distance among intra-cluster points and intersecting clusters. As a result, it would be perfect to test the robustness of our chosen clustering algorithms.

## 6. Experiment results

Using our 2 datasets, we conducted experiments on both clustering algorithms to test how well each of them performs on each dataset. Both algorithms have some drawbacks based on the dataset size, density and distribution. Our experiments are designed to uncover such flaws and to highlight the cases when the algorithms perform best.

### 6.1 DBscan

DBscan requires epsilon (eps) and min_samples values for its initialization. These are essentially the hyper-parameters that need tuning for each dataset. In the following experiments, we start by fitting a default DBscan model, tuning each hyper-parameter separately to gauge their effect on the accuracy and then combining them in a grid search approach to obtain the optimum set of parameters that give the best accuracy.

**Additional data processing**

1. **Accuracy:** While calculating the accuracy for each dataset, it is important to keep in mind that the number of clusters in DBscan can change. So we developed a function that maps the cluster numbers to our class labels during model fitting, which is then used to find out how many points were correctly classified. The number of correctly classified points can then be used to calculate the accuracy. The figure below details our output for a training sample-

```
CLUSTER: -1 ALLNUM: 161 CORRECT: 70 PRECISION: 0.4348 LABEL: 1
CLUSTER: 144 ALLNUM: 27 CORRECT: 27 PRECISION: 1.0000 LABEL: 3
CLUSTER: 91 ALLNUM: 22 CORRECT: 22 PRECISION: 1.0000 LABEL: 2
```

Fig. 19 - output parameters from DBscan clustering

We use the number of correct samples and precision score to predict the total accuracy of the model for later comparison.

2. **PCA:** Both datasets contain multiple attributes. As a result, PCA (Principal Component Analysis) dimension reduction has been applied to both to reduce their parameters to 2 for visualisation purposes.

#### 6.1.1 Seeds dataset experiments

The seeds dataset contains points with not much variation in their distances. This dataset follows a distribution ideal for DBscan, and therefore should obtain a good accuracy.

**Tuning min_samples -**

Our first experiment involves finding a relation between accuracy and the number of min_samples in DBscan initialization. The number of min_samples are varied while keeping the eps value constant (at 0.3575). The resultant plot can be seen below.

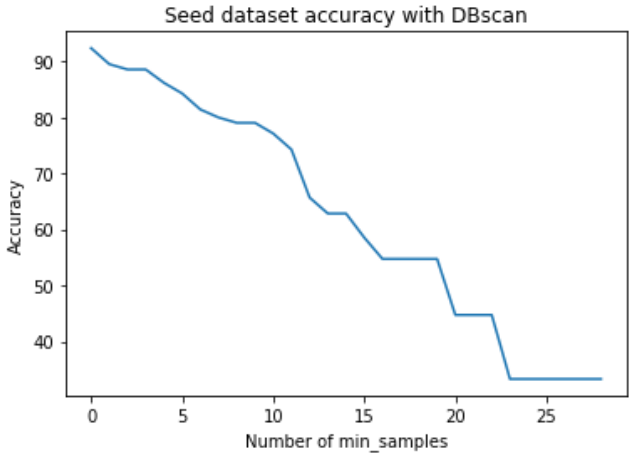


Fig. 20 - Accuracy v/s min_sample size for seeds dataset

From the plot above, we see a negative dependency of min_sample size with accuracy.

### Tuning eps value

This experiment involves finding the relationship between the epsilon value and the accuracy obtained by the DBscan model. We vary the eps value while keeping the min_samples constant (at 2).

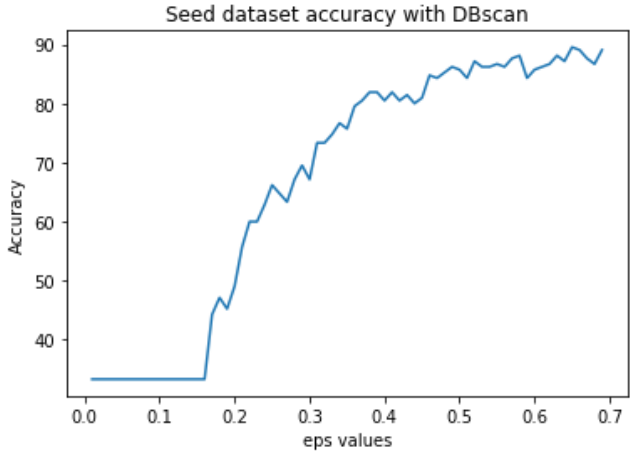


Fig. 21 - Accuracy v/s epsilon value for seeds dataset

From the plot above, we see a position correlation between epsilon value and the accuracy. This value starts to converge after a certain eps threshold.

### Grid search for eps and min_samples

It is difficult to judge the optimal values of the 2 hyper-parameters combined. As a result, we have adopted a grid search approach to find the optimal values for both parameters.
We define a range of eps values to test and for each eps value, we find the accuracy for each size of min_samples in our predefined range.
After obtaining all the accuracy values, we plot the heatmap given below.

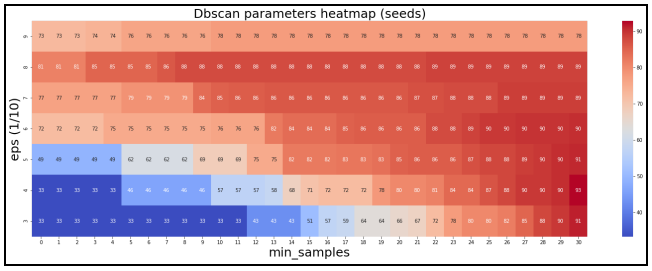


Fig. 22 - Accuracy heatmap for DBscan parameters (seeds)

From the heatmap above, we see the best accuracy is obtained with (**eps = 0.4, min_samples = 30**) with an **accuracy = 93.21%**. The clusters obtained by DBscan fit on these parameter values are given below.

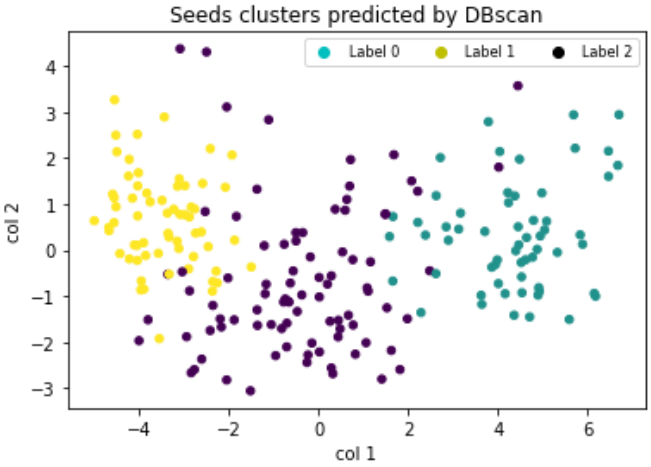


Fig. 23 - clusters obtained by DBscan on seeds dataset

When compared to the clusters in the original seeds dataset. This DBscan prediction comes close to modelling the actual clusters. Additionally, the number of optimal clusters chosen by DBscan were 3, which is also the number of class labels in the dataset. This is due to the fact that the data points can be generalised as a set of 3 separate clusters with relatively dense points.

However, some of the data points were assigned to an incorrect cluster. There were the points that mostly lay on the cluster boundaries with relatively less density, therefore introducing some ambiguity in the cluster prediction.

### 6.1.2 Forest dataset experiments

From the figure below, we can see the distribution of points in the forest dataset. This dataset was chosen to highlight the shortcomings of DBscan clustering, as this clustering method suffers when data points are associated with clusters of varying density (as is the case with forest dataset). Its data points do not form any well defined clusters and the point density reduces as we move from left to right.

### Tuning min_samples

This experiment is used to gauge the relation between number of min_samples and the clustering accuracy. The number of min_samples is varied from 1 - 30, while keeping the value of eps constant (at 0.25).

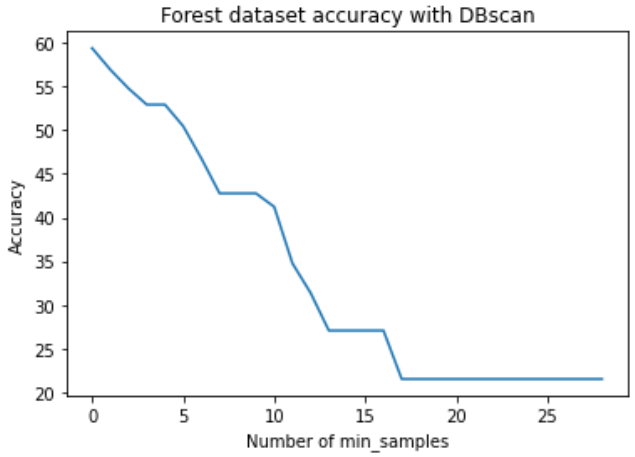


Fig. 24 - Accuracy v/s min_sample size for forest dataset

From the figure above, we can clearly observe an inverse relation between the 2 quantities. This behaviour was also observed with the seeds dataset and is reconfirmed.

### Tuning eps value

The diagram below represents the accuracy of DBscan clustering across the epsilon values for the forest dataset. It involves varying the eps value from 0.00 - 0.30 with increments of 0.01, while keeping the value of min_samples constant (at 2).

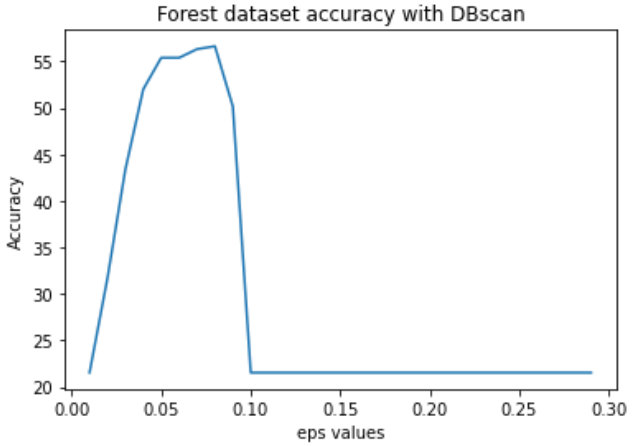


Fig. 25 - Accuracy v/s epsilon value for forest dataset

From the diagram, we can see that the data forms an almost concave graph with a maximum peak value of accuracy. The accuracy peaks to about ~56% when the eps value is set to 0.08.

### Grid search for eps and min_samples

Experimenting with different values from eps and min_samples, we were only able to achieve a maximum accuracy in the range 50%-60% when testing with each parameter individually. However, to obtain the combined optimal values of both parameters, a heatmap was constructed with varying values for both. The heatmap can be seen below.

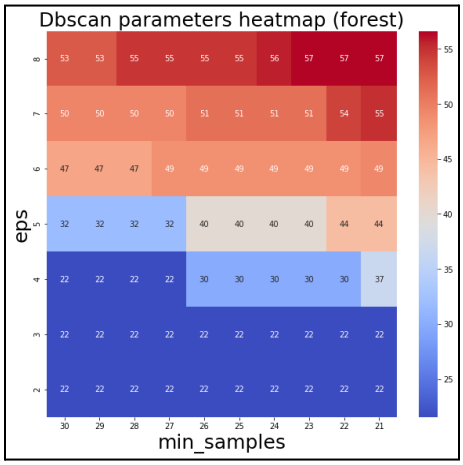


Fig. 26 - Accuracy heatmap for DBscan parameters (forest)

From the heatmap above, we see the best accuracy is obtained with (**eps = 0.08, min_samples = 23**) with an **accuracy = 57.06%**. The clusters obtained by DBscan fit on these parameter values are given below.

These ideal parameters were then used to plato the clusters for the forest dataset (given below). It is key to note that the number of clusters finally chosen by DBscan are 3, while the number of class labels in the dataset is 4. This can be attributed to the fact that the points of this dataset are not concisely partitioned into separable dense clusters which throws off the DBscan clustering.

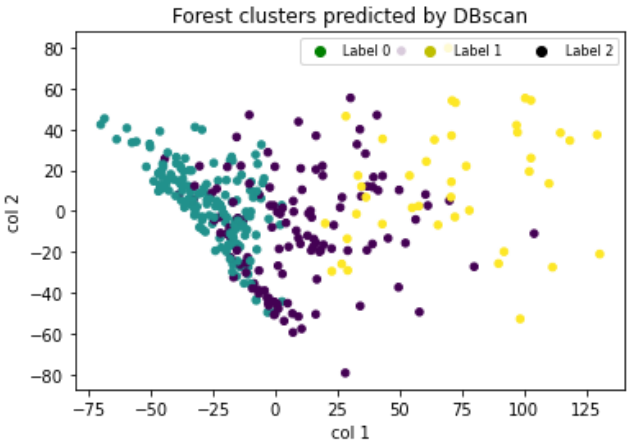


Fig. 27 - clusters obtained by DBscan on forest dataset

The model was only able to achieve an accuracy of 57.06 on this dataset. The primary reason for this low accuracy is that DBscan essentially merged the clusters for labels 0 and 2 in the original data. Therefore, half of these points were associated with the wrong cluster, adding to the error.

### 6.2 K-Means

We have used both the solutions discussed above in the explanation of K-Means. We initialized our clustering algorithm with ‘K-Means++’ and also used the elbow method to choose the optimal number of clusters.

We divided our data with a ratio of 70% train and 30% test. We first trained our KNN model on this train data with the number of clusters(‘K’) ranging from 1 to 9. We used inertia as an evaluation metric to determine the best value of ‘K’ keeping the computational cost in mind.

We evaluated the best K-Means model on the test data. We used a 'crosstab' to make a confusion matrix to show the clusters and the class the data point is of. We can calculate the accuracy by seeing the confusion matrix and taking the maximum amount for each cluster column as correctly identified data point. The confusion matrix is the results of the predictions of the model on the test data.

We can then calculate the accuracy by the formula: Accuracy = (Correctly identified labels) / (Total labels)

We later used PCA to perform dimensionality reduction to 2 features for better visualizations of the label points and cluster points. We used the test data to show the predicted clusters against the true labels of the data.

### 6.2.1 Seeds dataset experiments

The elbow curve for the train seeds data:

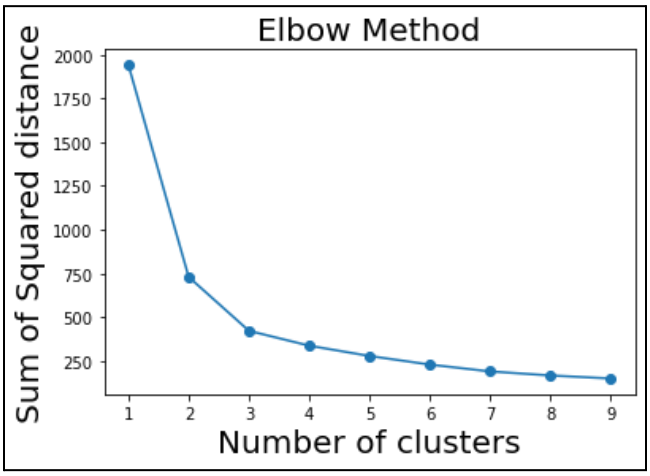


Fig. 28 - Elbow curve for seeds data

When we analyze the graph, we can see that the graph rapidly changes at a point of k=3 and thus creates an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value of 3 corresponding to this point is the optimal K value or an optimal number of clusters.

The following matrix is the results of the predictions of the model on the test data.

| cluster | 0 | 1 | 2 |
|---|---|---|---|
| label | | | |
| 1 | 16 | 0 | 4 |
| 2 | 4 | 17 | 0 |
| 3 | 1 | 0 | 21 |

Fig. 29 - Confusion matrix for clustering on seeds data

In the figure, we can see observe the following:

- cluster '0' has 16 as correctly identified points for label 1.
- cluster '1' has 17 as correctly identified points for label 2.
- cluster '2' has 21 as correctly identified points for label 3.
- In total, there are 63 data points.

The accuracy: ((16+17+21) / (63))*100 = **85.71%**

We visualize the original labels points and the identified clusters as follows:

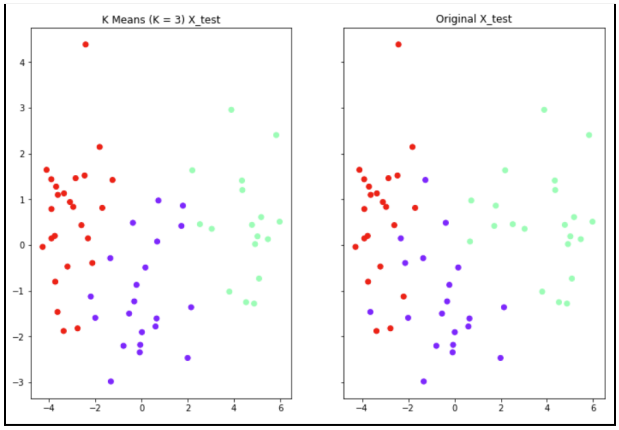


Fig. 30 - Visualize original data points and clusters formed

Here we can see that the clusters have identified the labels pretty well. We can also see the outliers being identified in a cluster which was one of the disadvantages of this algorithm. If used as a classification, we can say that the K-Means model has a good accuracy.

### 6.2.2 Forest data experiments

The elbow curve for the train forest data:

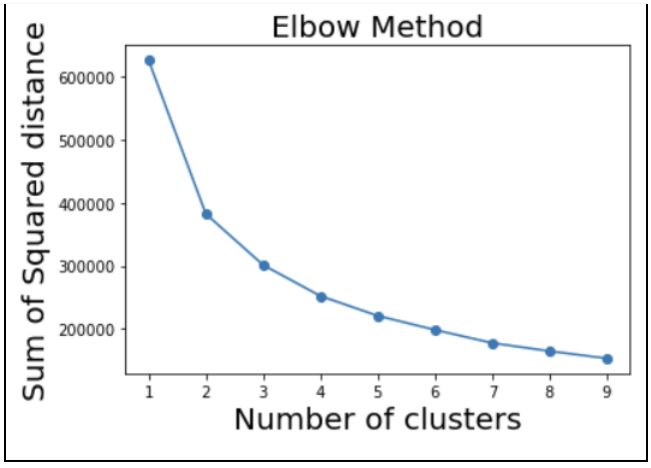


Fig. 31 - Elbow curve for forest data

When we analyze the graph, we can see that the graph rapidly changes at a point of k=4 and thus creates an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value of 4 corresponding to this point is the optimal K value or an optimal number of clusters.

When we analyze the graph, we can see that the graph rapidly changes at a point of k=3 and thus creates an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value of 3 corresponding to this point is the optimal K value or an optimal number of clusters.

The following matrix is the results of the predictions of the model on the test data.

| cluster label | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | 13 | 13 | 2 | 0 |
| 2 | 0 | 1 | 0 | 12 |
| 3 | 3 | 2 | 7 | 0 |
| 4 | 0 | 40 | 0 | 5 |

Fig. 32 - Confusion matrix for clustering on forest data

In the figure, we can see observe the following:

- cluster ‘0’ has 13 as correctly identified points for label 1.
- cluster ‘1’ has 40 as correctly identified points for label 4.
- cluster ‘2’ has 7 as correctly identified points for label 3.
- cluster ‘3’ has 12 as correctly identified points for label 2.
- In total, there are 98 data points.

The accuracy:((13+40+7+21)/(98))*100= **73.46%**

We visualize the original labels points and the identified clusters as follows:

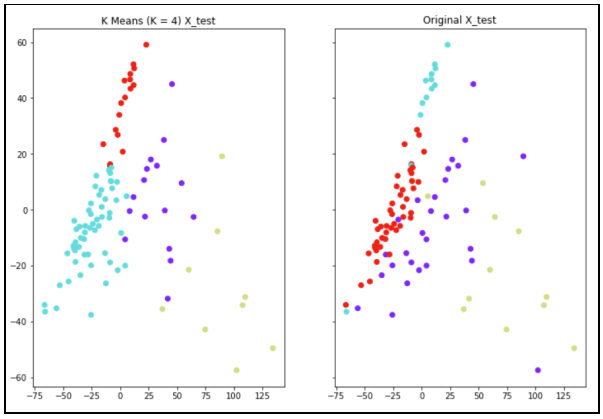


Fig. 33 - Visualize original data points and clusters formed

We can observe that the clusters change in every run which is the disadvantage of the K-Means algorithm. Thus, the colour coding of the Original and K-Means plots are different. Overall, we can comment that the clustering performance is good.

## 7. Comparative analysis

After tuning the hyper-parameters of both clustering models optimised on the 2 datasets, we get the following accuracy distribution-

| Dataset | DBscan accuracy (%) | K-Means accuracy (%) |
|---|---|---|
| Seeds | 93.21 | 85.71 |
| Forest | 57.06 | 73.46 |

It is clear from the above table that DBscan clustering outperforms K-Means on the seeds dataset. This difference in results can be explained by looking at the seeds data distribution in the figure below-

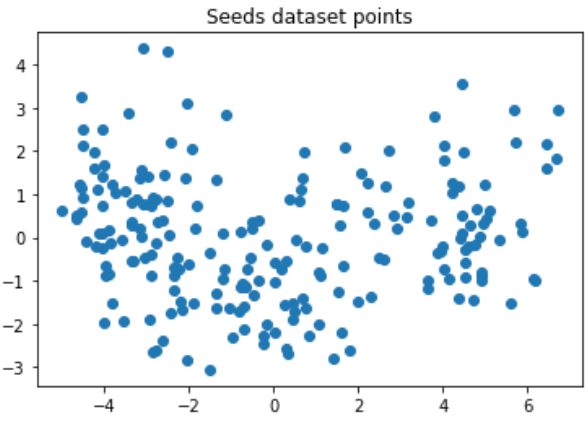


Fig. 34 - Distribution of seeds dataset points

The seeds dataset contains 3 clusters of relatively similar sizes and densities. Datasets of this distribution are ideal for DBscan as the combination of epsilon and min_samples remains constant throughout all clusters. As a result, DBscan outperforms K-Means.

However, the accuracy obtained for K-Means is much better than DBscan on the forest dataset. The data point distribution for forests can be seen in the figure below -

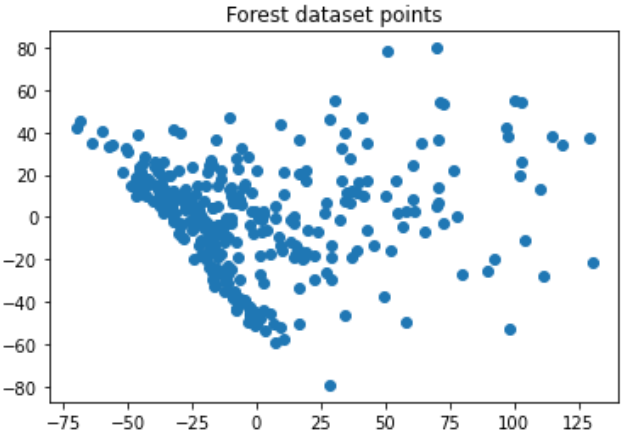


Fig. 34 - Distribution of forest dataset points

This data does not form any well defined clusters and lumps the data to the left, with varying densities to the right. Such datasets pose a problem for both algorithms, as no well-defined cluster boundaries would lead to cluster overlap resulting in incorrect cluster allocations. However, DBscan suffers more due to its reliance on epsilon value which relies on the density of the points. Since the density in the data varies highly, the overall accuracy drops further.

## 8. Conclusion

This study has detailed a series of experiments to obtain the numerical accuracies of 2 clustering algorithms (DBscan and K-Means) across 2 datasets with varying sizes and distributions. We have managed to successfully calculate the accuracies for both algorithms and used them for a comparative analysis.

Given the experiment results, it can be deduced that DBscan generally performs better than K-Means when the data is distributed into well-defined clusters with similar point densities. On the other hand, K-Means is better suited for datasets containing varying point densities and no perceivable cluster boundaries. Future works in this area can be extending our comparative approach to include other clustering algorithms and creating artificial datasets to gauge the accuracy on different distributions of datasets.

## 9. References

1. Dave, Meenu & Gianey, Hemant. (2016). Different clustering algorithms for Big Data analytics: A review. 328-333. 10.1109/SYSMART.2016.7894544.

2. Rodriguez MZ, Comin CH, Casanova D, Bruno OM, Amancio DR, et al. (2019) Clustering algorithms: A comparative approach. PLOS ONE 14(1): e0210236.

3. Farhad Pourkamali-Anaraki, Stephen Becker. (2016). Preconditioned Data Sparsification for Big Data with

Applications to PCA and K-Means. arXiv:1511.00152v3

4. Chen, Wen & Ji, Minhe & Wang, Jianmei. (2014). T-DBSCAN: A spatiotemporal density clustering for GPS trajectory segmentation. International Journal of Online Engineering (iJOE). 10. 19. 10.3991/ijoe.v10i6.3881.

5. Abhishek Sharma. (2020). Analytics Vidya: How to Master the Popular DBSCAN Clustering Algorithm for Machine Learning. Available- https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/