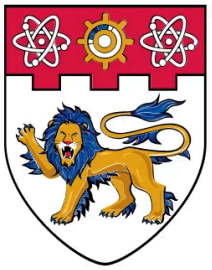

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4032 SC4020 Project 1

Members:

Poh Zi Jie Isaac
Dexter Voon Kai Xian
Lim Jun Yu
Mukherjee Tathagato

Table of Contents

1 Abstract

In this report, we explore and compare the performance of two popular clustering algorithms, K-Means and DBSCAN, across four datasets: diabetes, breast cancer, iris, and wine, all from the scikit-learn library. The goal is to understand how well each algorithm can find meaningful groupings within these datasets. For K-Means, the Elbow Method helped determine the optimal number of clusters, while DBSCAN required fine-tuning of its epsilon and min_samples parameters to work effectively. We used metrics like silhouette score, Adjusted Rand Index (ARI), and Adjusted Mutual Information (AMI) to evaluate the results. Overall, K-Means performed well on datasets with more structured, spherical clusters, whereas DBSCAN showed strength in handling clusters of different shapes but struggled with more complex, high-dimensional data. This analysis demonstrates that the choice of clustering algorithm depends heavily on the nature of the data.

2 Introduction

This report presents an in-depth analysis of two popular clustering algorithms, K-Means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), applied to four distinct datasets; the datasets used are the diabetes, breast cancer, iris, and wine dataset available from scikit-learn datasets module. The primary objective is to evaluate how these models perform in terms of identifying underlying patterns and groupings within the data.

Clustering, as an unsupervised learning technique, plays a crucial role in many data-driven tasks, such as customer segmentation, anomaly detection, and image processing. While K-Means relies on minimising variance within clusters and generally assumes spherical clusters of similar sizes, DBSCAN can detect clusters of arbitrary shape and identify noises, making it more flexible. By comparing these two algorithms across the chosen datasets, this report aims to highlight their strengths and limitations, providing insights into their suitability for different types of data distributions. The various parameters of each algorithm will be examined to draw conclusions on the effectiveness of the algorithm on the datasets. In addition, visual representations and performance metrics like the silhouette score will be used to quantitatively assess the quality of clusters formed by both algorithms. The findings aim to offer practical guidance on selecting appropriate clustering techniques for diverse real-world applications.

3 Methods

The first algorithm we will be using is the K-Means algorithm. The K-Means algorithm is used to partition a set of observations into K clusters where each observation belongs to the cluster with the nearest mean. The steps to implement the algorithm is: (1) specify the number k of clusters to assign, (2) randomly initialise the centroids, (3) assign each point to its closest centroid, and (4) compute the new centroid (mean) of each cluster. Repeat steps (3) to (4) until the centroid positions do not change. The optimal value of K can be determined by using the Elbow Method which looks at the change in the average distance to centroid as k increases. An extension of the K-Means algorithm is the K-Means++ algorithm which selects initial centroids that are far away from each other as compared to the K-Means

algorithm which selects centroids randomly. This improves the centroids initialisation which may lead to better clustering.

The second algorithm we will be using is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms. There are two key parameters for DBSCAN: (1) the epsilon_radius, and (2) the minPts. The epsilon_radius defines the epsilon-neighbourhoods; two points are considered neighbours if the distance between them is less than or equal to the epsilon_radius. MinPts is the minimum number of points to define a cluster. A data point can be identified as (1) core point, (2) border point or (3) outlier (noise point). A core point is a data point where there are at least minPts number of points in its surrounding area with radius eps. A border point is a data point that is not a core point but directly reachable from a core point. An outlier is a data point that is not a core point and not reachable from any core points. To implement the DBSCAN algorithm, firstly, we start a new cluster by randomly selecting a point x. If point x is a core point, start a new cluster from point x using step 2. Otherwise, label point x as noise point and repeat step 1. Secondly, we construct the cluster using breadth-first search and mark all the points in the cluster as visited. Repeat the steps until all points have been visited.

4 Experimental Analysis and Comparison

In this section we will focus on 4 different datasets for experimental analysis with K-Means and DBSCAN methods for comparison. Results from these experiments are shown in the respective section. In each dataset, we discuss the parameter settings and also their strengths and weaknesses of each method with respect to its dataset. Lastly, we will discuss and evaluate the key factors affecting the performance of the clustering methods.

4.1 Diabetes Dataset

4.1.1 K-Means on Diabetes Dataset

Before using the K-Means clustering algorithm, we applied the Principal Component Analysis (PCA) technique to reduce the dimensionality of the dataset.

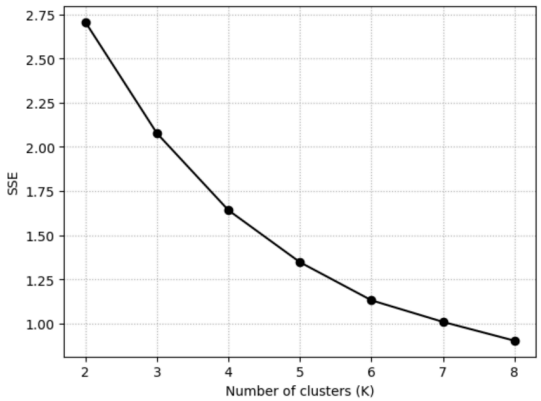

Fig. 1 SSE against Number of clusters(K) graph

From Fig. 1, we can observe that the Sum of Squared Errors(SSE) decreases as the number of clusters increases.

It is important to choose the appropriate number of clusters in K-Means as it will affect the quality of clustering. Choosing too few clusters can lead to underfitting, while choosing too many clusters can lead to overfitting. Therefore, we will obtain the optimal value of K through the elbow technique and find a good elbow point of K=3.

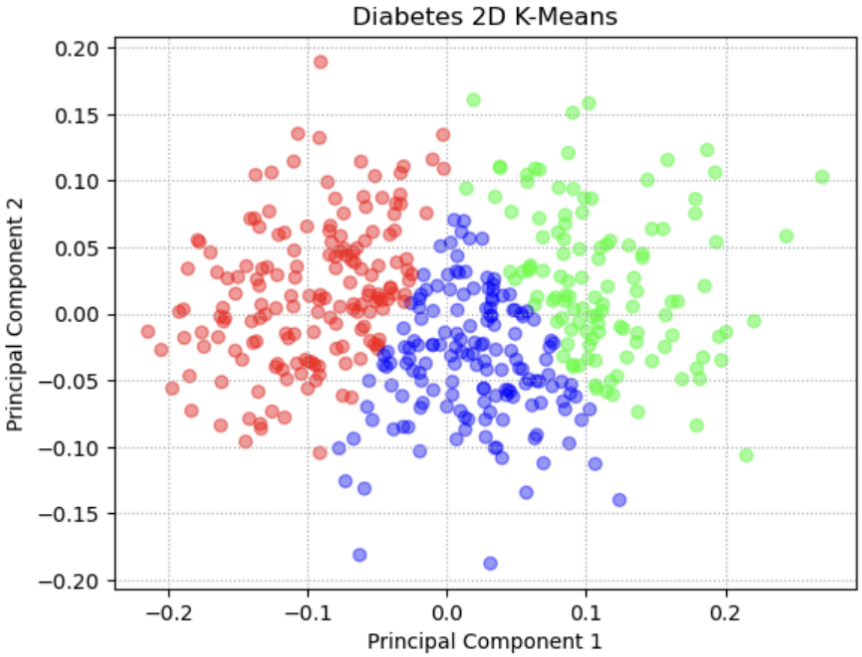

Fig. 2 K-Means Clustering with K=3

From Fig. 2, we observe that K-Means clustering produces non spherical clustering of the dataset. This may suggest that K-Means may not be a suitable method for clustering for the diabetes dataset. Furthermore, within the cluster, the points are widely spread out which means high variance within the clusters. Generally, the results for different initial centroids as shown from Fig. 3 are relatively consistent which indicates stability and robustness of the diabetes data.

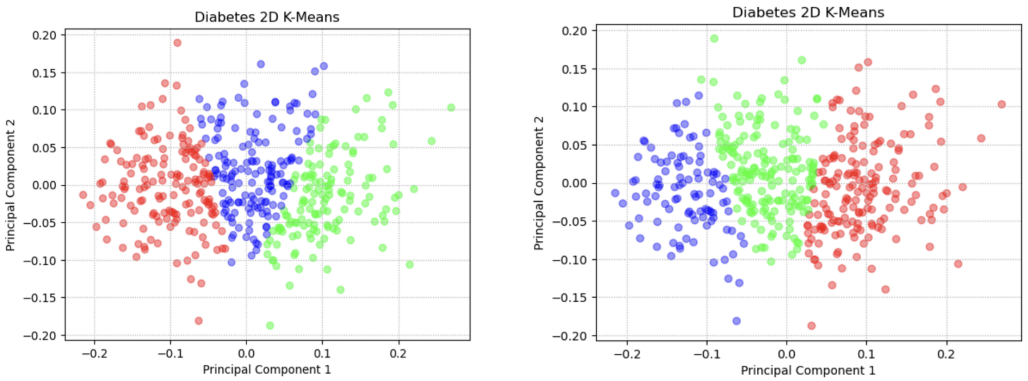

Fig. 3 KMeans Clustering with K=3 (Different initial centroids)

From Fig. 4, we observe that the Silhouette Score of the diabetes dataset is about 0.33, which is a moderate cluster separation and is not particularly strong.This may imply that while some clusters are somewhat distinct, there may be overlap or ambiguity between them.

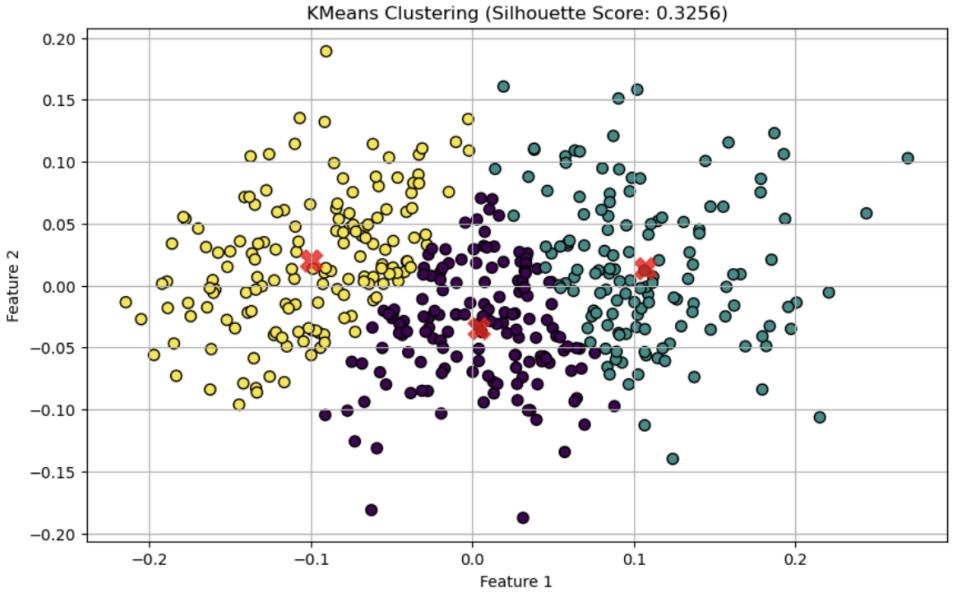

Fig. 4 KMeans Clustering (Silhouette Score: 0.3256)

4.1.2 DBSCAN on Diabetes Dataset

However, while we perform DBSCAN on the diabetes dataset, we observe that this method encounters a situation where DBSCAN identifies a single cluster or labels all points as noise. This could be due to the nature of the dataset where the data does not have enough points in dense regions to meet the min_samples threshold and the clusters are too far apart which results in DBSCAN labelling many points as noise.

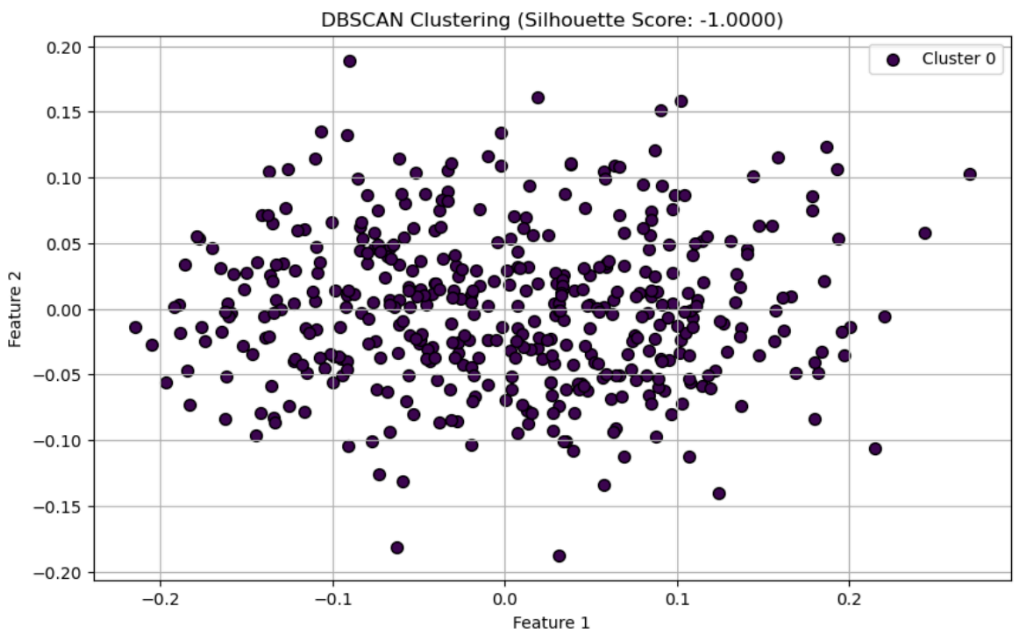

Fig. 5 DBSCAN Clustering (Silhouette Score: -1.0000)

Furthermore, the inherent structure of the data can also affect clustering. As DBSCAN is a density based approach, the data distribution results in a differing silhouette score as shown from Fig. 5. Therefore, the DBSCAN method in this scenario may suggest poor clustering and excessive noise present in the dataset.

4.1.3 Comparison between K-Means and DBSCAN for Diabetes Dataset

We can observe that K-Means clustering provides a better silhouette score of 0.33 whereas our DBSCAN method was unable to produce good clustering results due to the structural property of the dataset of diabetes. The impact is significant because the diabetes dataset has ten dimensions, making it highly dimensional.

4.2 Breast Cancer Dataset

This dataset has a total of 10 different attributes, of which the mean, standard deviation and largest value of these attributes were computed for each image, thus a total of 30 features. However for simplicity, only the mean attributes are used for clustering.

All the 10 features used are normalised to take on values between 0 to 1. This ensures that the clustering algorithms will not disproportionately assign greater weights to features of higher magnitude.

Next, principal component analysis is adopted to decrease the number of features used in the models. This allows a greater level of explainability and saves a large amount of memory loading and processing. From the cumulative variance ratio in Fig. 6, it shows that it is possible to retain almost 100% of the variation by using only 3 out of the 10 features. Thus, 3 components will be used. The features are then fitted and transformed to apply dimensionality reduction.

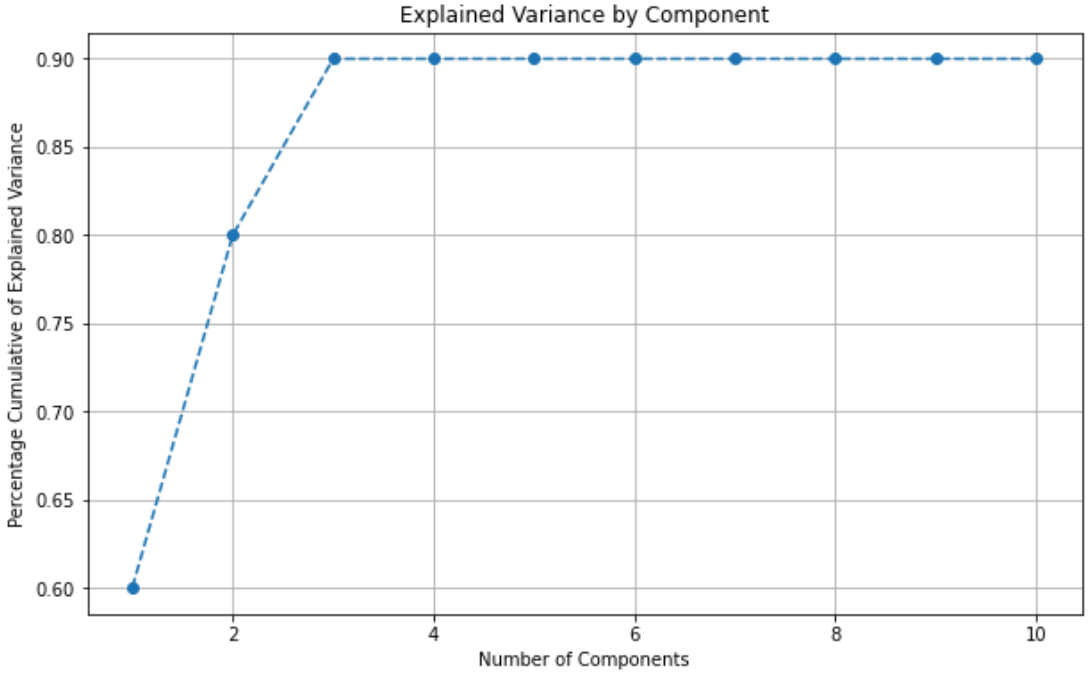

Fig. 6 Cumulative variance ratio

4.2.1 K-Means on Breast Cancer Dataset

Firstly, K-Means++ algorithm is used, where the initialization of the centroid is such that each centroid is as far from the other as possible. To determine the ideal number of centroids, a total of 28 iterations are run, from 1 cluster to 28. The average distance to centroid for each iteration is plotted against the number of clusters initialised. Based on the elbow method, we can see from Fig. 7 that the ideal number of clusters is 5.

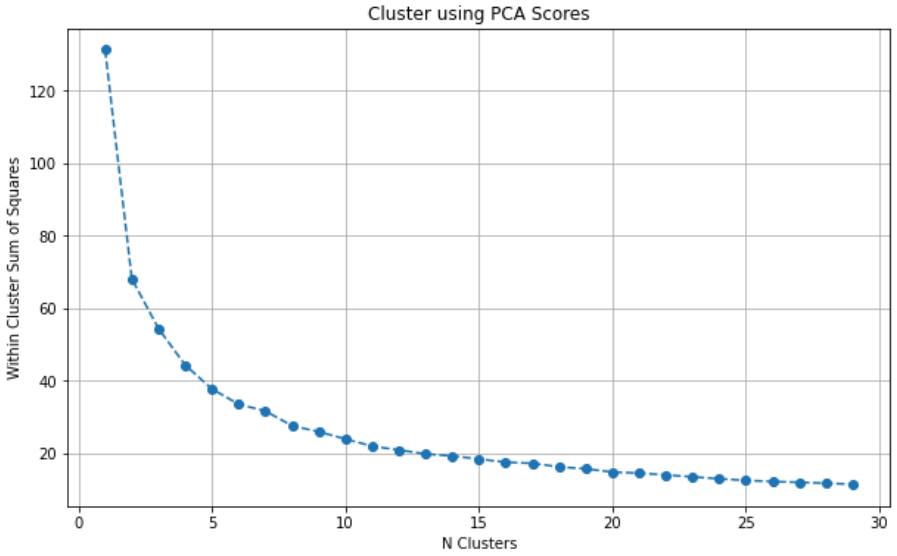

Fig. 7 Within Cluster Sum of Squares values to number of clusters to determine ideal number of clusters

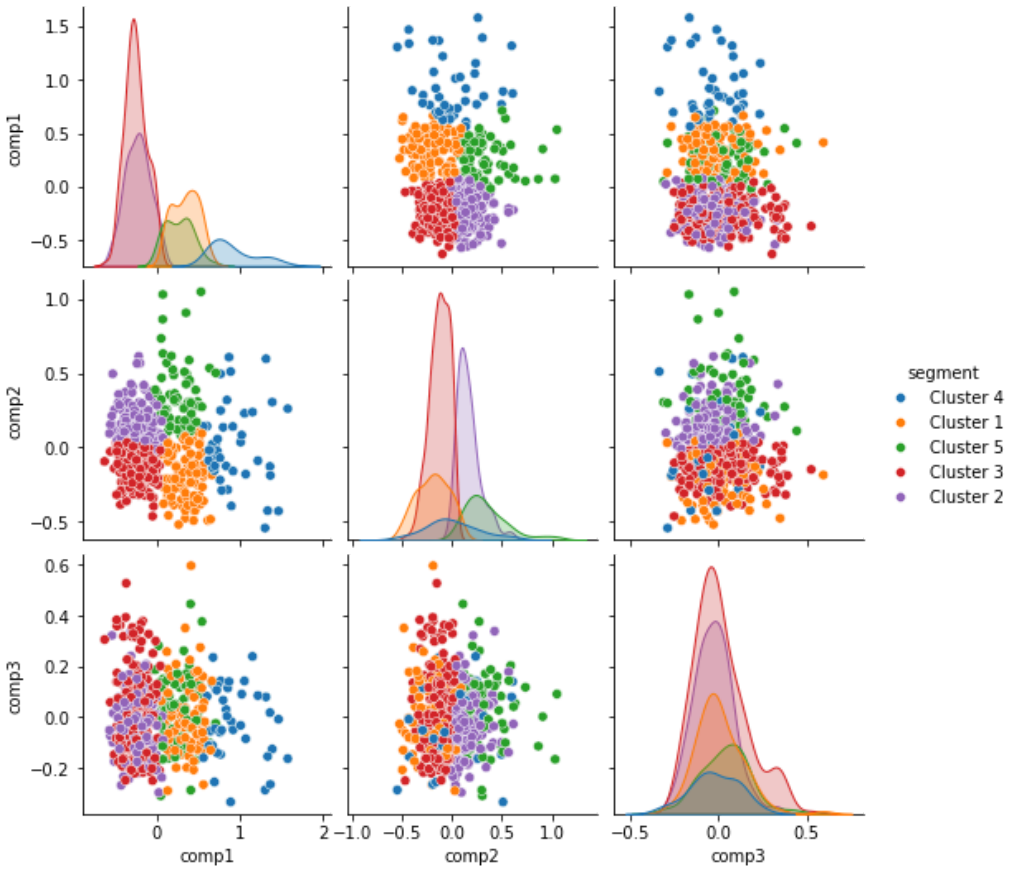

Fig. 8 Pairplot of each component with K-Means++

From the correlation plots shown in Fig. 8 of each component, we see some well-defined patterns of clusters between component 1 and 2. Note that the amplitude of the red cluster is the highest in all cases, suggesting a higher probability for a new observation to be located within it. The silhouette score for the above model using KMeans++ is 0.27390, which is relatively low.

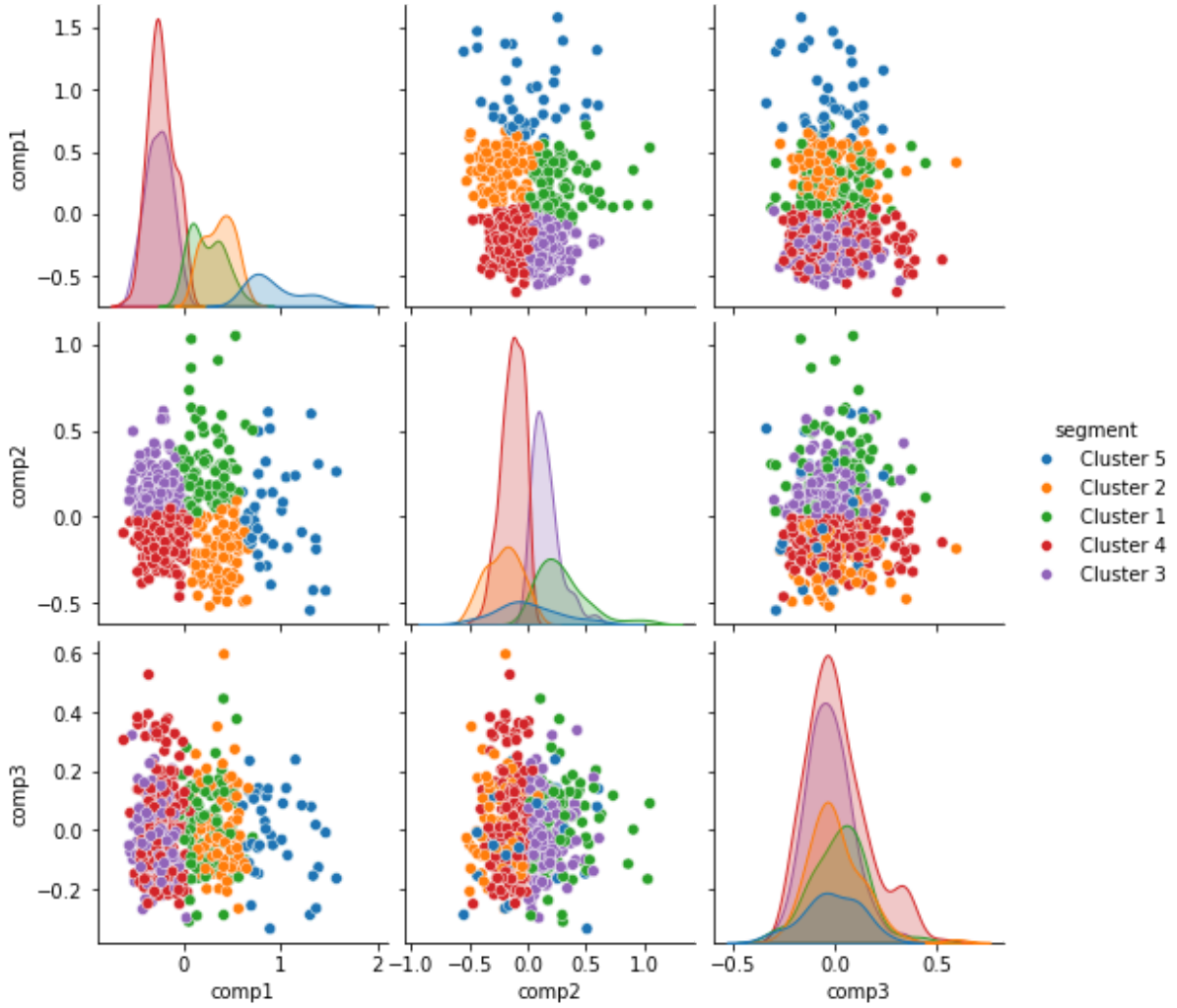

Fig. 9 Pairplot of each component with K-Means

The above pairplot is for another model where the initialisation of centroids is random. We obtained a silhouette score of 0.261234, very similar to the previous model. Both models of K-Means performed as badly on the dataset.

This dataset is commonly used for binary classification. However, the ideal number of clusters found was 5, which is not binary. Nevertheless, it might not be entirely incorrect since benign and malignant cancers have different grades on their own as well, which might be the reason why more clusters were identified.

4.2.2 DBSCAN on Breast Cancer Dataset

Next method used for evaluation is DBSCAN. To choose the minimum number of points in a cluster, a general rule of thumb is to select it to be equal or higher than the number of dimensions. In the following model, the minimum number of points to determine a cluster shall be 6.

Next, to determine the ideal radius *epsilon.* We used the nearest neighbour algorithm based on the minimum number of points to find out the ideal *epsilon*. The results are plotted below in Fig. 10. Once again, the elbow method determines that the *epsilon* is approximately 0.125.

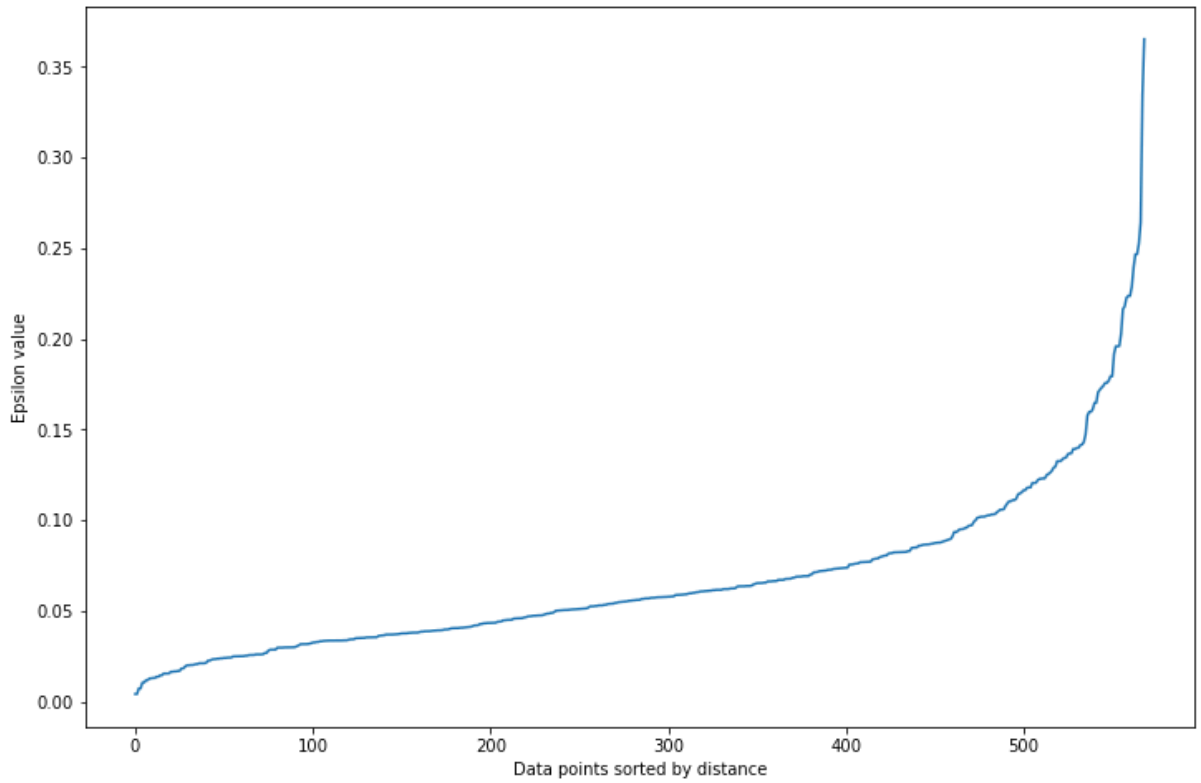

Fig. 10 Epsilon value against data points to determine the ideal epsilon value

After fitting the data with the DBSCAN model, a silhouette score of 0.59112 is obtained and a total of 2 clusters were identified, excluding outliers. According to the score, the DBSCAN model could capture approximately 60% of the data.

4.2.3 Comparison between K-Means and DBSCAN for Breast Cancer Dataset

Comparing the two models, we can see that DBSCAN has done significantly better than K-Means. Apart from the higher silhouette score, it also correctly identified the number of different labels within the dataset. This might imply that K-Means is not as suitable for high dimensional datasets.

4.3 Iris Dataset

The Iris dataset is a multivariate dataset used and made famous by the British statistician and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems*. The data set consists of 150 samples of flowers with 50 samples from the three Iris species: (1) Iris setosa, (2) Iris virginica, and (3) Iris versicolor. Four features were measured for each sample: (1) Sepal length, (2) Sepal width, (3) Petal length and (4) Petal width. This dataset can be imported from the `sklearn` package using `from sklearn.datasets import load_iris`.

4.3.1 K-Means on Iris Dataset

We determine the optimal number of clusters for the Iris dataset by using the Elbow Method which plots the Within-Cluster Sum of Square (WCSS) for different values of `k` and identifying the point where the rate of decrease sharply slows down known as the “elbow” point.

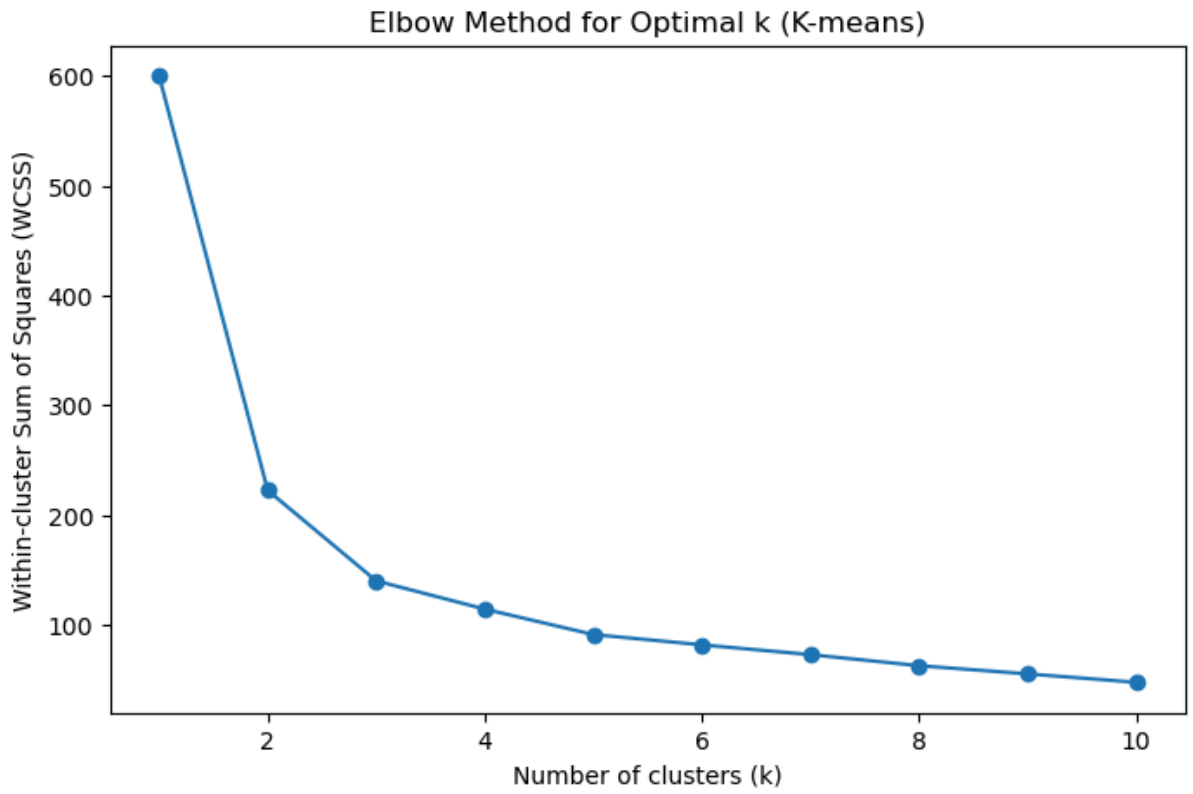

Fig. 11: Elbow Method for Optimal k

From Fig. 11, we can see that the “elbow” point occurs when `k=3`. Hence, the optimal number of clusters is 3. Next, we will perform K-Means clustering and K-Means++ clustering on the Iris dataset with `k=3`. Silhouette score will be used to determine the quality of clustering.

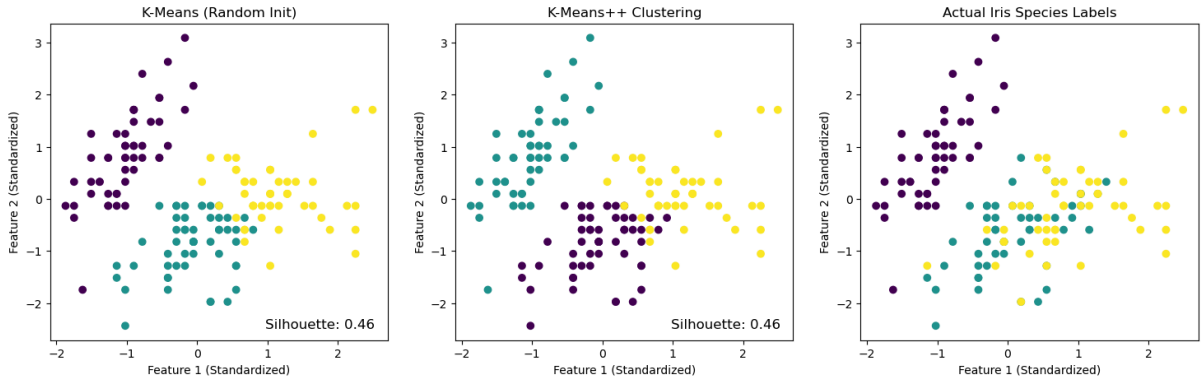

Fig. 12: Clustering results of K-Means and K-Means++ vs Actual labels

From Fig. 12, we can see that K-Means and K-Means++ does a relatively good job at clustering the data with a Silhouette Score of 0.46 for both methods. Both K-Means and K-Means++ can quite accurately identify the Iris setosa species, but both are having some difficulties in separating the vericolor species from the virginica species due to the overlapping of some of their features.

4.3.2 DBSCAN on Iris Dataset

As a rule of thumb, the minimum samples (MinPts) for DBSCAN should be greater than or equal to the dimensionality of the data set. Since the Iris dataset has four dimensions, we choose `MinPts=5`. We will determine the value of epsilon(eps) using k-nearest neighbours (KNN) method.

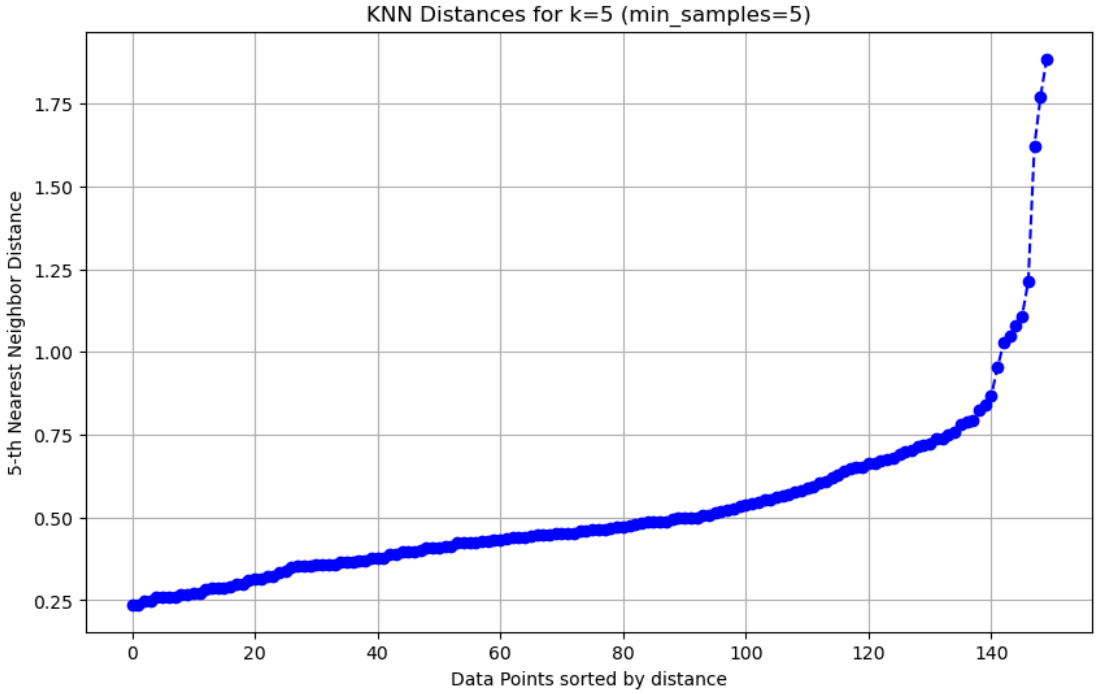

Fig. 13: KNN Plot to determine eps

From Fig. 13, we can see the elbow point occurs when eps is around 1.1, hence, we choose `eps=1.1`. As per K-Means algorithm, we will use Silhouette score to determine the quality of clustering.

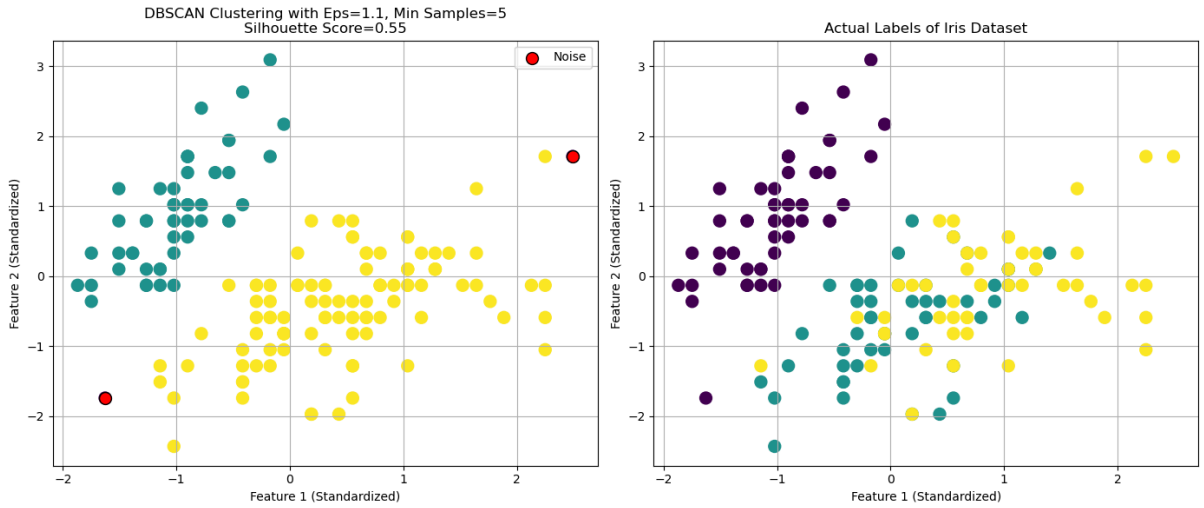

Fig. 14: Clustering results of DBSCAN vs Actual labels

From Fig. 14, we can see that it is able to quite accurately identify the setosa species, but it clustered the virginica species and versicolor species into one species. The silhouette score is 0.55 which represents quite good clustering. However, the clustering is not quite accurate, possibly due to the close proximity of the virginica and versicolor species. We can also observe that two points are labelled as noise/outliers which will not be used for clustering.

4.3.3 Comparison between K-Means and DBSCAN for Iris Dataset

Firstly, we can see that K-Means is able to identify the three species whereas DBSCAN only identifies two species. This is because DBSCAN identifies clusters based on density and some of the points of the virginica species and versicolor species are in close proximity.

Secondly, we see that the silhouette score for DBSCAN is 0.55 which is higher than the silhouette score of 0.46 for K-Means. The higher silhouette score for DBSCAN can be misleading as the clustering results do not align with the actual labels.

Lastly, we can see that DBSCAN identifies noise/outlier points whereas K-Means does not. The presence of noise/outlier points might affect the clustering results of K-Means.

4.4 Wine Dataset

This dataset is found on the scikit-learn library on python, and was sourced from the University of California Irvine Machine Learning site. The dataset consists of 13 features and 178 observations of data, classifying types of wine. Due to the high dimensionality of the data, direct visualisation is impossible. In order to visualise the data, Principal Component Analysis (PCA) is used to reduce the dataset to two dimensions. PCA allows us to represent the data points in a 2D space, preserving the highest variance possible and enabling visual inspection of the data's structure. A visualisation of the data projected into two principal components is shown in Fig. 15.

When attributes are unscaled, the clusters seem to overlap at times, and clustering algorithms like K-Means and DSCAN are sensitive to the scale of the features. To ensure fair

comparison and avoid biases caused by different feature magnitudes, the data was standardised using **StandardScaler()**. This method subtracts the mean and divides by the standard deviation for each feature, ensuring that all variables are on the same scale and contribute equally to the clustering process. The standardised data will be used for all subsequent clustering analysis. A visualisation of the scaled dataset can be seen in Fig. 15.

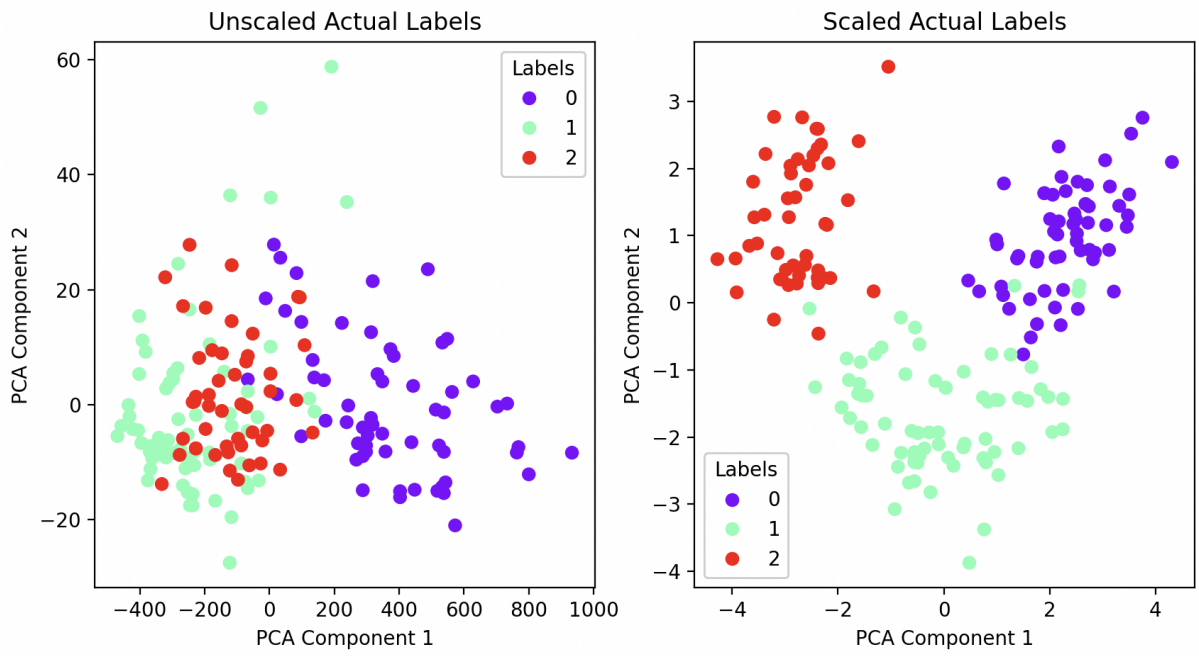

Fig. 15: Visualisation of Wine Dataset, unscaled (left) and scaled (right) after PCA

4.4.1 K-Means on Wine Dataset

The optimal number of clusters in K-Means was determined using the Elbow Method, which evaluates the inertia (the sum of squared distances between data points and their closest cluster centre). By plotting inertia against different values of K, an “elbow” point indicates the optimal number of clusters. For this dataset, the elbow was observed at K=3, as shown in Fig. 16.

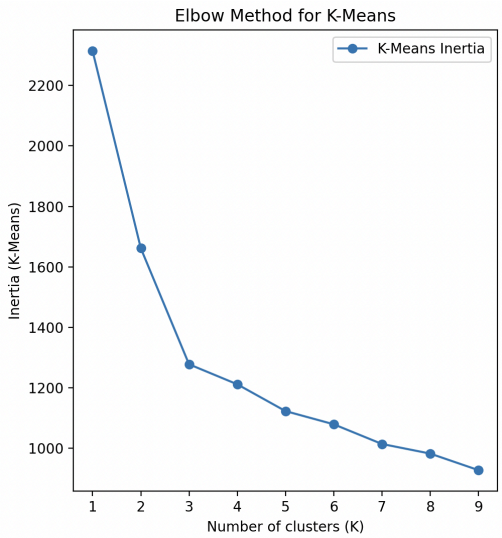

Fig. 16: Elbow method for K-Means, showing Inertia leveling off after K=3

Upon clustering, the graph of clusters is plotted, yet again with scaled attributes and PCA. This is compared to actual labels in Fig. 17.

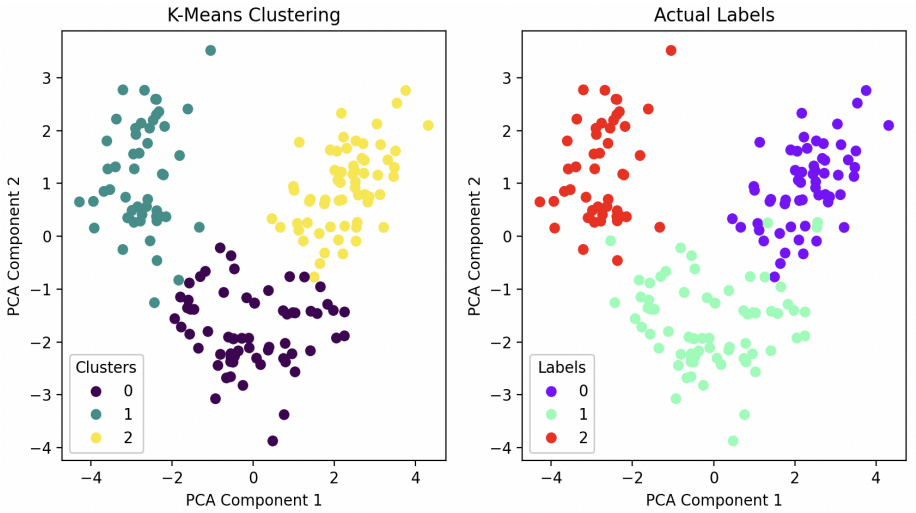

Fig. 17: K-Means clusters compared to Actual Labels, silhouette score = 0.284859

To evaluate the clustering results quantitatively, the silhouette score is used. The silhouette score quantifies the quality of clustering by measuring how similar an object is to its own cluster compared to other clusters. For K-Means clustering with K=3 clusters, the silhouette score was **0.284859**, indicating that the clusters are reasonably well separated, but improvements are still possible. In Fig. 17, it is noticeable that the boundaries of clusters are well defined using K-Means, but in the actual dataset, there are a few points that overlap between clusters.

4.4.2 DBSCAN on Wine Dataset

Initially, DBSCAN was applied using baseline parameters: **epsilon = 0.5** and **min_samples = 5**. Under these conditions, the algorithm failed to form meaningful clusters and identified all points as noise. The silhouette score for this configuration was **-1**, confirming that the clusters were not well-defined, as shown in Fig. 18.

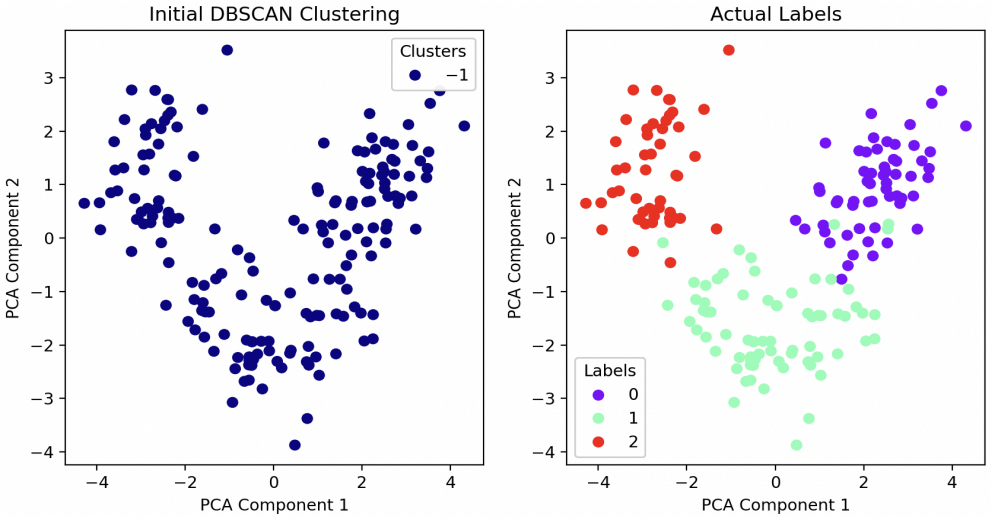

Fig. 18: Initial DBSCAN results, silhouette score = -1 (noise)

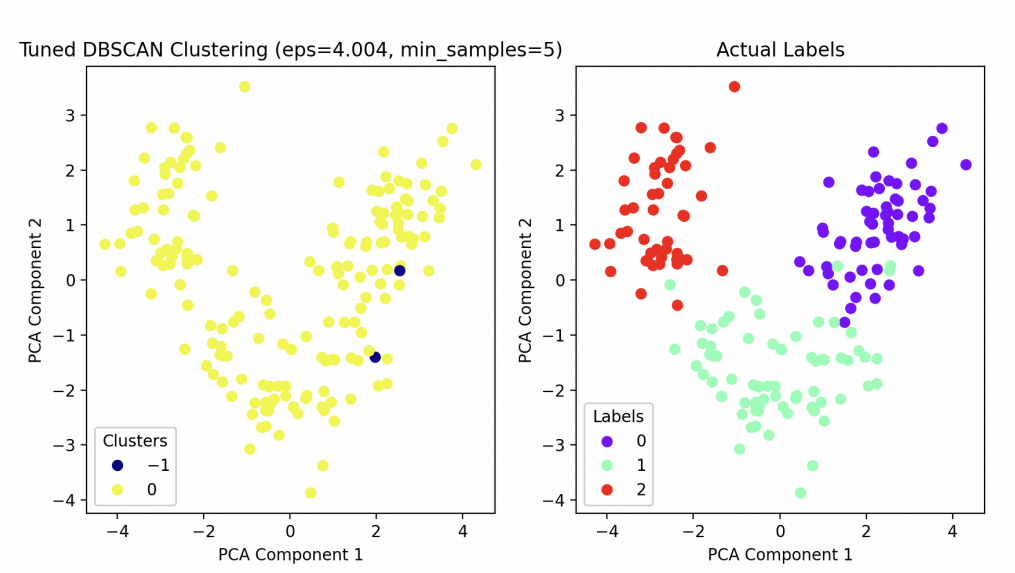

Fig. 19: Tuned DBSCAN results, silhouette score = 0.255925

To improve DBSCAN's performance, a parameter search was conducted, varying **epsilon** and **min_samples**. Through grid search, values of **epsilon = 4.0** and **min_samples = 5** were found to produce improved clusters. The corresponding silhouette score after tuning was **0.255925**, i.e. similar to that of K-Means. However, as indicated by Fig. 17 (K-Means) Fig. 19, the DBSCAN clusters were far from the K-Means results, and even further from the ground truth. This implies the need for other metrics to measure clustering quality.

In addition to the silhouette score, Adjusted Rand Index (ARI) and Adjusted Mutual Information (AMI) were used to assess the quality of DBSCAN's clustering:

- **ARI**: Measures the similarity between the predicted clusters and the true labels, adjusted for chance. The ARI for the tuned DBSCAN was **-0.003691**.
- **AMI**: Evaluates the mutual information between the predicted clusters and the true labels, accounting for chance. The AMI for the tuned DBSCAN was **0.005846**.

As the ARI and AMI scores for DBSCAN are near zero (even though the silhouette score is comparable to that of K-Means), DBSCAN clusters do not provide much accuracy or information.

4.4.3 Comparison between K-Means and DBSCAN for Wine Dataset

A comparison of the clustering methods was conducted using three metrics: **silhouette score**, **Adjusted Rand Index (ARI)**, and **Adjusted Mutual Information (AMI)**. The results are summarised in Table 1 below.

Clustering Method	Silhouette Score	ARI	AMI
K-Means	0.284859	0.897495	0.874579
DBSCAN (Tuned)	0.255925	–0.003691	0.005846

Table 1: Summary of Results

The silhouette score of **0.28** for K-Means indicates that the clusters are moderately well-separated but not perfect. The silhouette score for DBSCAN, **0.26**, is slightly lower than K-Means, indicating that DBSCAN's clusters are less well-separated than those of K-Means. DBSCAN is particularly sensitive to noise, and this lower silhouette score suggests that the algorithm may have struggled to define clear clusters given the dataset structure or parameter tuning.

The ARI and AMI scores for K-Means are very close to 1, indicating that the clustering results are highly aligned with the true labels. The same scores for DBSCAN are close to zero, implying that DBSCAN's clusters do not align with the actual labels at all. DBSCAN has essentially failed to capture any meaningful cluster structure based on the true labels, potentially indicating that DBSCAN's assumption of density-based clusters does not align well with the distribution of this dataset or that more extensive tuning is required.

5 Conclusion Analysis

In conclusion, both K-Means and DBSCAN are widely used algorithms, each with its unique strengths and weaknesses. The choice between them largely depends on the characteristics of the dataset and the specific goals of the analysis.

K-Means is particularly effective for datasets where clusters are expected to be spherical and of similar sizes. Its simplicity and speed make it suitable for large datasets, where it can quickly converge to a solution. However, K-Means assumes that clusters are compact and equally sized, which can lead to suboptimal results in scenarios where the data contains clusters of varying shapes and densities. Furthermore, K-Means is sensitive to outliers, as it tends to incorporate them into clusters, potentially skewing results.

On the other hand, DBSCAN excels in identifying clusters of arbitrary shapes and is robust against noise and outliers. Its density-based approach allows it to find meaningful clusters in datasets with varying densities. DBSCAN does not require the number of clusters to be specified a priori, making it flexible for exploratory analysis.

However, it requires careful tuning of parameters such as eps and min_samples, and may struggle with datasets that contain clusters of similar density or are very high-dimensional. Our example datasets for wine and diabetes, which have 10 and 13 dimensions, respectively, demonstrate these findings.

Conversely, for datasets with complex shapes, varying densities, or significant noise, DBSCAN is likely to yield more meaningful results, which can be seen from our example datasets for iris and breast cancer.

In summary, the effectiveness of K-Means and DBSCAN can vary significantly depending on the dataset. For datasets with well-defined, spherical clusters, K-Means may be the preferred choice due to its speed and simplicity.

Ultimately, we should consider the specific characteristics of our datasets, possibly applying both methods to gain comprehensive insights, and utilise metrics such as the Silhouette score to assess clustering quality. By doing so, we can then make informed decisions about the most appropriate clustering technique for their analysis needs.

6 References

1. Jecksom, J. (2023, April 11). *Clustering and Principal Component Analysis (PCA) from Sklearn*. Medium. https://medium.com/@jackiee.jecksom/clustering-and-principal-component-analysis-pca-from-sklearn-c8ea5fed6648
2. Stack Abuse. (n.d.). *DBSCAN with Scikit-Learn in Python*. Stack Abuse. Retrieved October 12, 2024, from https://stackabuse.com/dbscan-with-scikit-learn-in-python/
3. Bhattacharyya, S. (2019, June 9). *DBSCAN ALGORITHM: Complete Guide and application with Python Scikit-Learn*. Medium. https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d
4. Arvai, K. (n.d.). *K-means clustering in Python: A practical guide*. Real Python. https://realpython.com/k-means-clustering-python/
5. *Load_diabetes*. scikit. (n.d.). https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.load_diabetes.html
6. *Load_iris*. scikit. (n.d.-b). https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.load_iris.html
7. *Load_wine*. scikit. (n.d.-c). https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.load_wine.html
8. *Load_breast_cancer*. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html