Investigating the Configurable Parameters of K-means Unsupervised Learning

Research question: To what extent is the performance of the k-means clustering algorithm in unsupervised learning influenced by the initial placement algorithm, the number of features, and the number of clusters?

Word count: 3998

Contents

1 Introduction **4**

2 Background Information **5**

2.1 Supervised and Unsupervised Learning 5

2.2 K-Means Clustering 5

2.2.1 Parameters 7

2.2.2 First configurable parameter: initial placement 7

2.2.3 Second configurable parameter: number of clusters 8

2.2.4 Third configurable parameter: number of features 8

2.2.5 Feature Scaling 8

2.2.6 Dimensionality Reduction Through principal Comoponent Analysis (PCA) 9

3 Methodology **11**

3.1 Data Sets Used 11

3.1.1 Synthetic data set 11

3.1.2 Wine data set 12

3.2 Evaluation metrics 12

3.2.1 Silhouette score 12

4 Experimental results **14**

4.1 Table of Synthetic data set Results 14

4.2 Table of Wine data set Results 15

4.3 Example of Programmed Outcome 15

4.4 Graphical Presentation of Achieved Results 17

4.4.1 Graphical presentation of synthetic data set results 18

4.4.2 Graphical presentation of wine data set results 19

5 Data Analysis **20**

5.1 Analyzing Number of Clusters Using Silhouette Score 20
5.2 Analyzing Initialization Methods 21
5.3 Analyzing the Number of Features 22
5.3.1 Importance of Feature Scaling 22
5.3.2 No Direct Relationship Between Features and Clusters 22
5.3.3 Analysis of Dimensionality Reduction Using PCA 23

6 Limitations **23**

7 Conclusion **24**

1 Introduction

As our world continues to technologically advance, machine learning has taken on a role of propelling the future. One fascinating machine learning technique is clustering (also known as cluster analysis). Clustering is a process of discovering patterns in unlabeled data (data that has not been tagged with identification [1]), and aims to group individual objects based on their degree of similarity from one another [2]. Clustering can be applied to many aspects of the real world: from grouping customers based on their behavioral psychology to grouping different types of wine (a data set that will be explored in this investigation) - clustering results are influential in all disciplines.

Within clustering algorithms, there are many configurable parameters that affect the overall performance. It is vital to understand the differences in performance of each algorithm when certain parameters are customized in order to maximize the effectiveness of clustering for a given unlabeled data set.

This paper seeks to evaluate the effectiveness in performance, measured through silhouette score and the number of iterations of three configurable properties of k-means clustering (i.e., initial placement, number of clusters, number of features). K-means clustering serves to extract value from large unlabeled data sets. This gives the results of this research the potential to improve the efficiency of clustering applications. By understanding how certain parameters of k-means clustering can be configured to maximize the effectiveness, industries such as business would benefit greatly from better client, product, and data clustering for their operations.
The following research question will be explored: **To what extent is the performance of the k-means clustering algorithm in unsupervised learning influenced by the initial placement algorithm, the number of features, and the number of clusters?** For this investigation, k-means clustering algorithms were programmed to group data from a synthetic data set and a public wine data set. For each data set, the initial placement,

number of clusters, and number of iterations were altered at each rerun. Patterns were analyzed and performance was evaluated through calculated silhouette scores and the number of iterations needed to complete the process. This investigation will also determine whether the metric used, silhouette score, is a reliable determinant of accuracy of an unsupervised learning algorithm. Logical and mathematical explanations for the results obtained are discussed.

2 Background Information

2.1 Supervised and Unsupervised Learning

Machine learning algorithms have two main approaches - supervised and unsupervised learning. Supervised learning algorithms refers to working with labeled data sets to train and "supervise" algorithms in processing data. Since input and output data are labeled, the supervised learning model can easily measure accuracy. Classification and regression algorithms are the most common types trained by supervised learning, due to their nature of reliance on a labeled data set [3].

Unsupervised learning discovers hidden patterns without the need of human interaction or labeled data sets. The main tasks associated with unsupervised learning are clustering, association, and dimensionality reduction.

2.2 K-Means Clustering

This paper will specifically explore the k-means clustering algorithm, a method of vector quantization that originally stems from signal processing.

Given a set of observations (x_1, x_2, ..., x_n), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq$ n) sets S = S_1, S_2, ..., S_k so as to minimize the within-cluster sum of squares (i.e., variance).

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Since k-means clustering is computationally difficult [4], a heuristic iterative refinement technique was introduced, in which k is pre-defined prior to the clustering process. In this investigation, the number of clusters k will be one of the three configurable parameters.

The approach k-means follows to solve the problem is called Expectation-Maximization. The E-step (expectation) is assigning the data points to the closest cluster. The M-step (maximization) is computing the centroid of each cluster. Here is a rundown of how k-means operates:

1. Specify number of clusters k.
2. Initialize centroids by first shuffling the data set and then randomly selecting k data points for the centroids without replacement.
3. Keep iterating until all stopping criteria are met. As k-means is an iterative process, it is crucial to understand when to stop the algorithm. Essentially, the three stopping criteria are when centroids of newly formed clusters do not change, when points remain in the same cluster, and when the maximum number of iterations is reached [5].

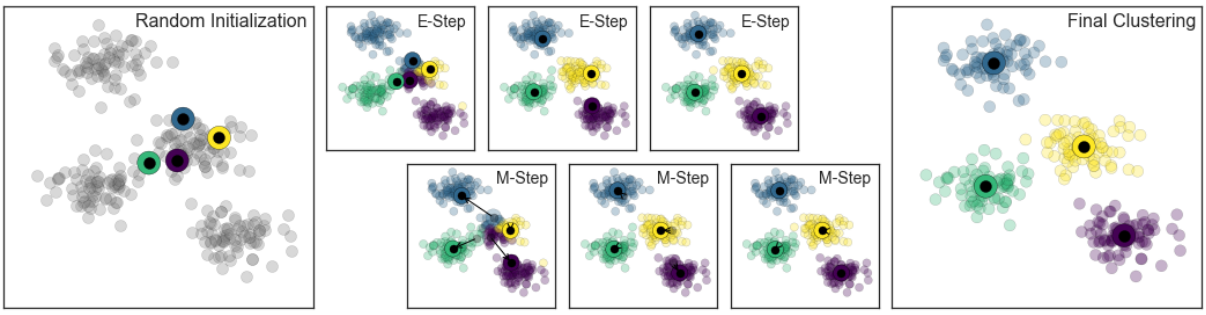


Figure 1: Example of unlabelled data going through the k-means Expectation-Maximization process [6]

2.2.1 Parameters

In this paper, the effects of three different parameters on the performance of k-means processes will be investigated across two data sets - a synthetic data set and a real data set containing the chemical properties of certain wine types.

2.2.2 First configurable parameter: initial placement

The first altered parameter will be the initial placement, which will be set to either random or k-means++. This refers to the initial placement of the clusters in the k-means clustering process. A random initial placement means that the center-points of clusters are randomly chosen. K-means++ is a biased random sampling that chooses centers farther apart from one another, avoiding close points; it aims to achieve the optimal clustering results in a fewer number of iterations. The first chosen centroid of k-means++ is random, and the next centroids are chosen as the datapoints with the largest squared distance from the first chosen centroid.

(a) Distance from each centroid

(b) Blue datapoint is chosen as third centroid

Figure 2: Determining third centroid [5]

The above figure demonstrates the use of k-means++ to determine the third centroid of a set of datapoints. The square of the distances of each datapoint from its closest centroid (green or red) is calculated, and the blue datapoint is selected as the third centroid since it has the largest squared distance from its nearest centroid in Figure 2a.

2.2.3 Second configurable parameter: number of clusters

The second configurable parameter in this investigation is the number of clusters. There is no limit to how many clusters can be formed in k-means clustering. We will be determining the optimum combination of three configured parameters with the synthetic and wine data sets in this investigation.

2.2.4 Third configurable parameter: number of features

The final parameter to be configured in this investigation is the number of features. The number of features can vary greatly for real world data sets. In the context of students at a school, features include nationality, gender, grades, household income, etc. This is an interesting area of exploration, since on the surface level it may seem that more features makes it easier to find similarities and establish clusters. However, it could also be harder because the conditions of similarity become much more nuanced.

2.2.5 Feature Scaling

Feature scaling is an important step to take prior to processing data for many machine learning algorithms. It is implemented through standardization, which re-scales the features to reflect the properties of a standard normal distribution. This is vital in many algorithms as they may behave badly if individual features do not represent normally distributed data. For example, if an investigation aims to describe the physical attributes people and the data provided includes their heights in centimeters and weights in pounds, a five pound difference cannot directly be compared to a five centimeter difference in height.

Features are standardized by removing the mean and scaling to unit variance. As an example, the standard score of a sample x is calculated as: $z = (x - u)/s$, where u is the mean of the training samples and the variable s is the standard deviation of the training samples.

2.2.6 Dimensionality Reduction Through principal Comoponent Analysis (PCA)

One instance feature scaling is used is during Principal Component Analysis, or PCA. PCA is a dimensionality reduction method typically used to reduce the feature dimensionality of large data sets. This is done by transforming a data set with many variables into a smaller one with less variables but is still able to capture most of the information of the original large data set. Simply put, the goal of dimensionality reduction methods such as PCA is to decrease the number of variables of a data set while preserving as much information as possible [7].

To better understand PCA, refer to the graph below, Figure 3. There are 10 principal components seen, meaning the original data set is 10-dimensional, having 10 features/-variables. principal components are essentially crafted as combinations of all ten of the variables. They are mixed in such a way that most information of variables is compressed into the first few principal components (as represented by the highest percentage of explained variances being in principal component 1).

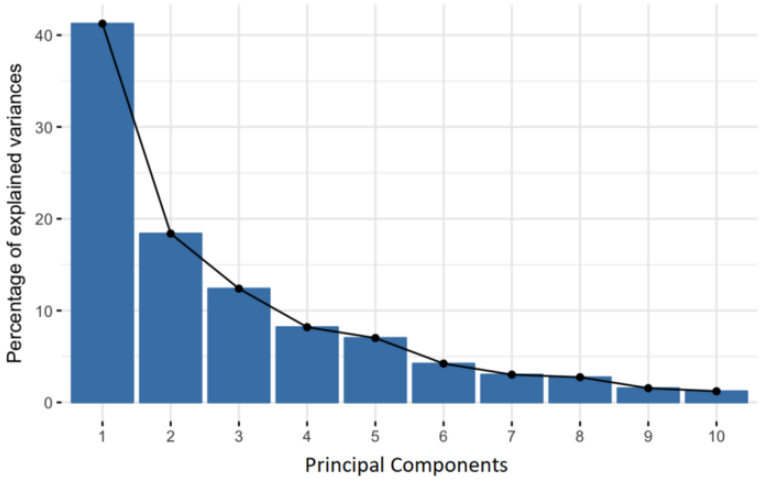


Figure 3: Principal component analysis [7]

One obvious issue of lowering the number of variables in the data set is that accuracy will be negatively affected. However, the intent of dimensionality reduction methods are to sacrifice a little accuracy for simplicity. This is because smaller data sets without extraneous variables are easier to investigate, making the visualization and analyzing processes of machine learning algorithms much easier, faster, and more streamlined.

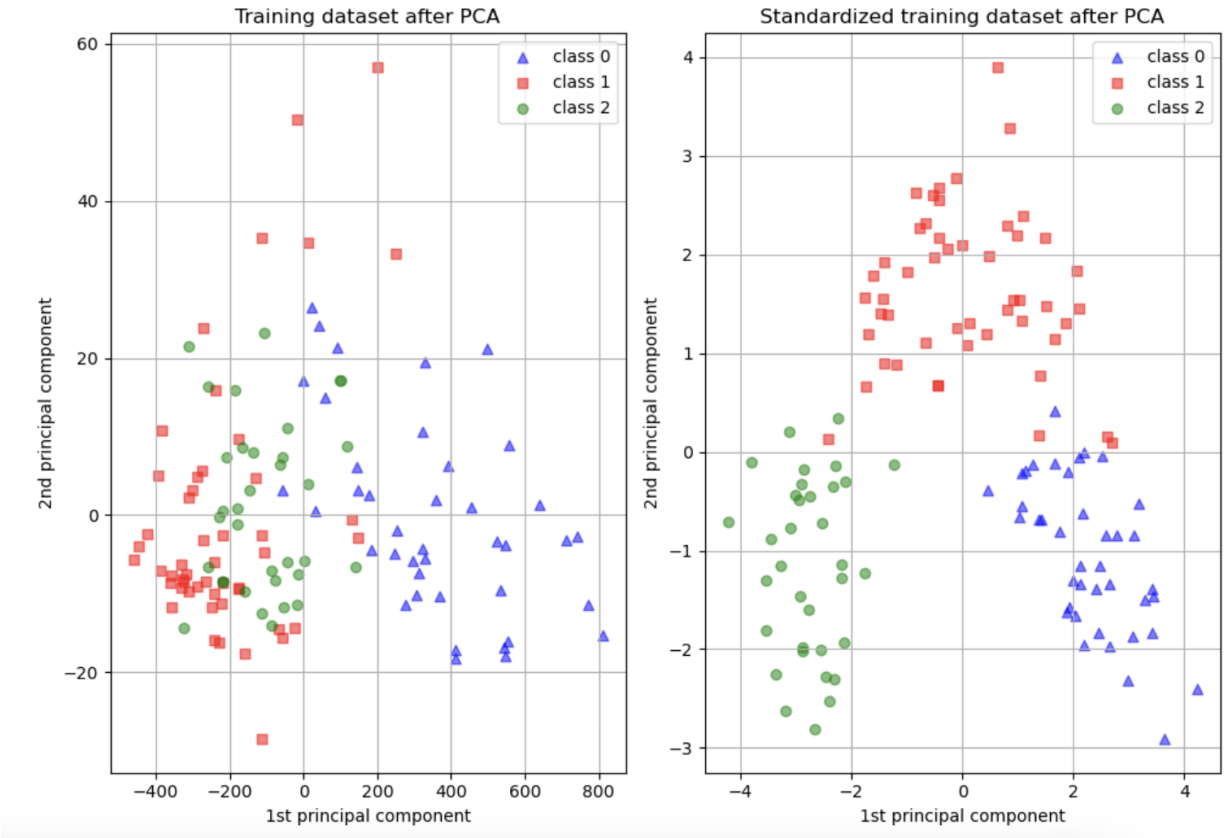


Figure 4: Visualization of feature scaling through principal component analysis [8]

As mentioned in the previous subsection, feature scaling standardizes the features of a data set. In order to effectively execute PCA, feature scaling is required in order to better compare variables and determine which to reduce. In Figure 4 above, there are supposedly three classes, and the machine learning algorithm is aiming to cluster these three classes accordingly. If successful, there should be a clear distinction between the three classes (represented by three different colors and shapes). An example of PCA without feature scaling is shown on the left, with two principal components (data set variables) selected on the x and y axes. Each entity that belongs to different classes are

mixed together, demonstrating a failed attempt at clustering. However, after undergoing feature scaling, as shown on the right, the PCA proves to be much more effective. There is a clear distinction that can be seen between entities of each of the three classes. The feature-scaled version on the right greatly outperforms the non-scaled version on the left, emphasizing the importance of feature scaling in dimensionality reduction through PCA.

3 Methodology

Primary experimental data is the main source of data in this paper. Two data sets (a synthetic and a wine data set) were used to complete a k-means clustering process (code in appendix, adapted from an example from Scikit-learn [9]). The number of iterations taken to run each configured program was recorded and accuracy was displayed by silhouette score. This investigation took an experimental approach because there was limited secondary data to answer the research question. The chosen approach allows independent variables to be easily manipulated. Since an experimental approach was taken, the results of the experiment are technically limited to the scope of the procedure.

The hardware configuration used was an Apple MacBook Air (M1, 2020) with 16GB Memory. The software package used in the code was `Python 3.9.0 and scikit-learn 1.1.1`.

3.1 Data Sets Used

3.1.1 Synthetic data set

The synthetic data set used in this investigation generates the sample data from the make_blobs Python function. This particular setting has one distinct cluster and 3 clusters placed close together. Below is the source code used that generates the synthetic data set.

```
X, y = make_blobs(
    n_samples=1000,
    n_features= 20,
    centers=4,
    cluster_std=1,
    center_box=(-10.0, 10.0),
    shuffle=True,
    random_state=1,
)
```

3.1.2 Wine data set

The wine data set includes 3 classes, with each class containing 59, 71, and 48 samples, respectively. For each row, there are 13 real and positive features. The 13 features are Alcohol, Malic acid, Alkalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines, and Proline [10].

	alcohol	malic_acid	ash	...	hue	od280/od315_of_diluted_wines	proline
0	14.23	1.71	2.43	...	1.04	3.92	1065.0
1	13.20	1.78	2.14	...	1.05	3.40	1050.0
2	13.16	2.36	2.67	...	1.03	3.17	1185.0
3	14.37	1.95	2.50	...	0.86	3.45	1480.0
4	13.24	2.59	2.87	...	1.04	2.93	735.0

Figure 5: First five rows of wine data set

3.2 Evaluation metrics

3.2.1 Silhouette score

The metric of accuracy used in this paper is silhouette score, mainly used to evaluate the quality of clusters created. Silhouette score is calculated at each data point, and requires the mean distance between the observation point and all other data points in the same cluster. This is known as mean intra-cluster distance. In the following equation, the mean

intra-cluster distance is denoted by a, the mean nearest-cluster distance is denoted by b and the Silhouette score denoted by S.

$$S = \frac{(b-a)}{max(a,b)}$$

The range of silhouette scores is between -1 and 1. A score of 1 means that the cluster is itself dense and well-separated from other clusters. A value of 0 represents overlapping clusters, with their samples extremely close to the boundary of neighboring clusters. A negative score indicates inaccuracy, suggesting that the datapoints may have been assigned to the wrong cluster [11].

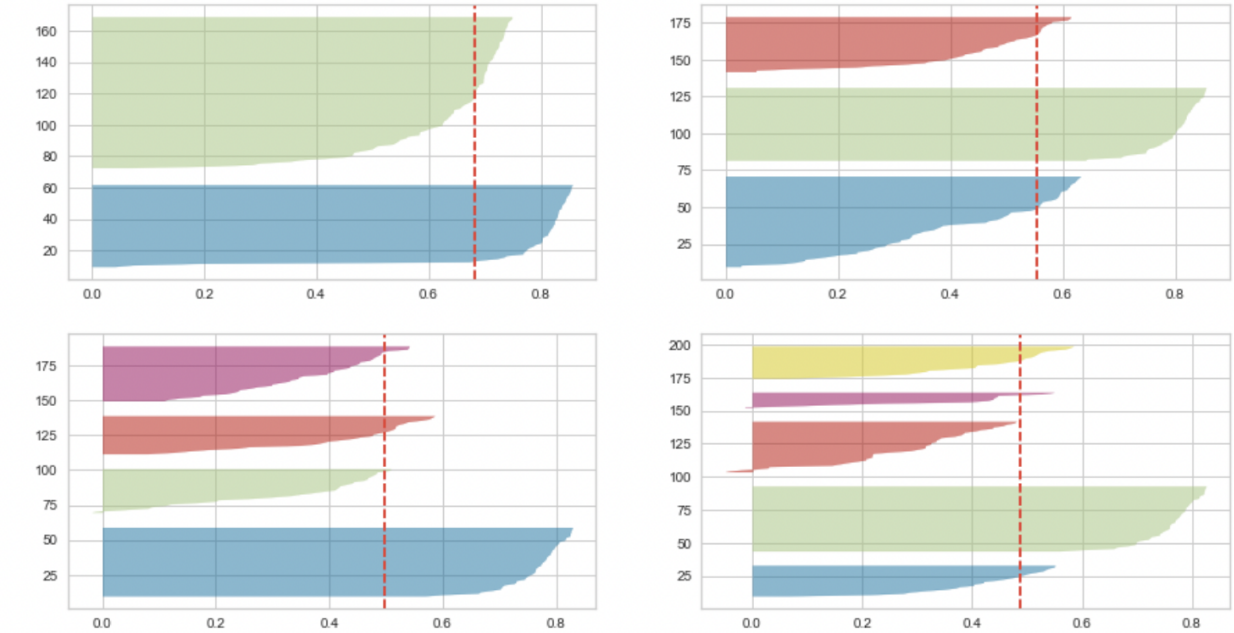


Figure 6: Example of silhouette analysis for 2,3,4,5 clusters [11]

Above is a visualization of Silhouette analysis done on 2, 3, 4, and 5 clusters. The average silhouette score is indicated by the vertical dotted line. The Silhouette scores of clusters 4 and 5 are sub-optimal for the given data set due to the presence of clusters with silhouette scores that are below-average and wide fluctuations in the size of the silhouette plots.

The silhouette score values for clusters 2 and 3 look relatively optimal. The score for each cluster is above the average silhouette score and there is minimal fluctuation in

size.

Silhouette plots with clusters having uniform thicknesses is an indication of the optimal number of clusters. The top right plot with 3 clusters have the most uniform thicknesses out of all four plots. Thus, the optimal number of clusters in the above figure is 3.

4 Experimental results

4.1 Table of Synthetic data set Results

The following table displays the experimental results of k-means clustering using the synthetic data set. Silhouette scores are displayed to four significant figures in order to maintain high accuracy, as differences between some values were rather minimal.

Number of clusters	Initial placement	Silhouette Score				Number of iterations			
		Number of features				Number of features			
		5	10	15	20	5	10	15	20
2	Random	0.4864	0.5067	0.5106	0.4780	4	2	2	3
	K-means++	0.4864	0.5067	0.5106	0.4780	4	2	2	3
3	Random	0.6583	0.6972	0.7137	0.6321	5	2	3	3
	K-means++	0.6583	0.6972	0.7137	0.6321	3	2	2	2
4	Random	0.7801	0.8145	0.7920	0.8328	2	3	3	2
	K-means++	0.7801	0.8145	0.7920	0.8328	2	2	2	2
5	Random	0.6182	0.6381	0.599	0.6378	14	12	10	15
	K-means++	0.6175	0.6242	0.6184	0.6355	7	17	14	15
6	Random	0.4610	0.4630	0.4245	0.4397	14	13	20	15
	K-means++	0.4607	0.4369	0.4455	0.4396	18	14	8	13

Figure 7: Synthetic data set experiment results

4.2 Table of Wine data set Results

The following table displays the experimental results of k-means clustering using the wine data set.

Number of clusters	Initial placement	Silhouette Score				Number of iterations			
		Number of features				Number of features			
		5	10	15	20	5	10	15	20
2	Random	0.3145	0.2373	0.2655	0.2837	11	9	10	14
	K-means++	0.3145	0.2373	0.2655	0.2837	4	11	4	11
3	Random	0.3590	0.2585	0.2339	0.2594	8	7	8	12
	K-means++	0.3590	0.2585	0.2333	0.2596	5	5	6	5
4	Random	0.3508	0.2413	0.2243	0.2071	8	14	6	10
	K-means++	0.3508	0.2493	0.2289	0.2217	8	12	10	7
5	Random	0.3121	0.2321	0.1744	0.1834	8	15	16	12
	K-means++	0.3256	0.2293	0.2008	0.1827	9	11	8	6
6	Random	0.3149	0.2445	0.1750	0.1540	14	7	11	12
	K-means++	0.3168	0.2338	0.1773	0.1983	10	6	7	9

Figure 8: Wine data set experiment results

4.3 Example of Programmed Outcome

In order to visualize some results of the code, the following two figures are the produced charts of the program. Figure 9 depicts the synthetic data set results with k-means++ initial placement, 5 features, and 4 clusters. The results depicted are optimal, since all four clusters in the left chart are almost of equal size and all cross the average silhouette score threshold indicated by the dotted red line. The right chart is a display of of the actual data-points being formed into clusters represented in the four colors that correspond with the left chart. It is vital to note that the right chart has x and y axes that represent two out of the five total features, as it is not easy to create a five-featured visual representation of the clusters. However, by comparing two features, the user can still clearly note the distinction between clusters. Figure 10 is not optimal, with no consistently sized clusters and only two clusters crossing the average silhouette score threshold.

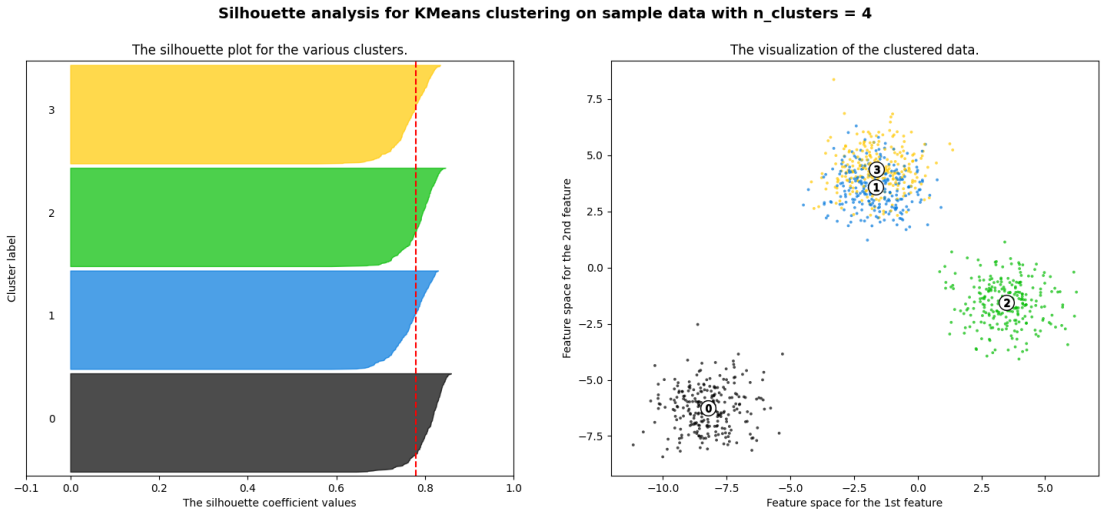


Figure 9: Synthetic data set results of k-means++ initialization, 5 features, and 4 clusters (figure generated by author)

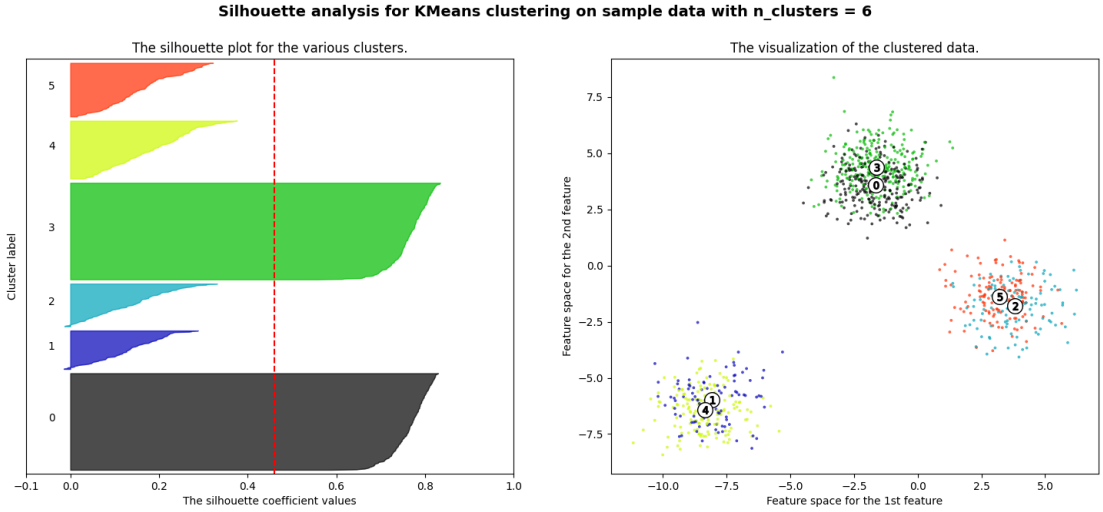


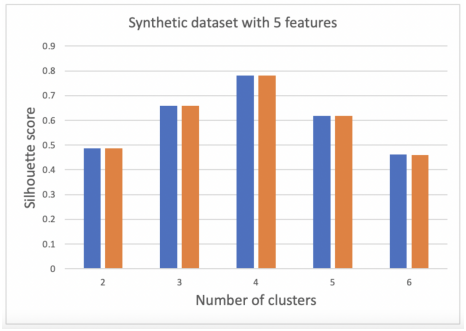
Figure 10: Synthetic data set results of k-means++ initialization, 5 features, and 6 clusters (figure generated by author)

4.4 Graphical Presentation of Achieved Results

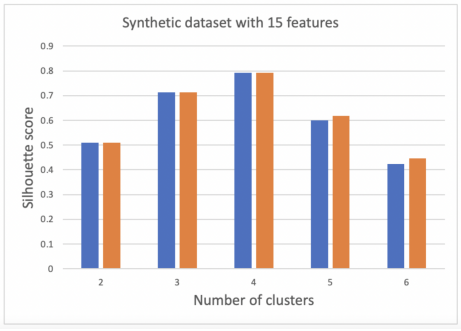
For ease of visualization, the data has been displayed in the bar charts below. In all bar charts, the left bar (blue) has random initial placement, and the right bar (orange or grey) has k-means++ initial placement.

The first row of the blue-orange charts displays the silhouette score of the data sets with 5 features (a) and 15 features (b). Each bar represents the results of a particular number of clusters (x-axis) on the accuracy (indicated by silhouette score on the y-axis) of the k-means clustering process. The second row is the same, except the y-axis is now replaced by the number of iterations. The final row (of blue-grey charts) display the silhouette scores (a) and number of iterations (b) of a data set with a set amount of clusters when altering the number of features (x-axis). The aforementioned row descriptions apply to the 14 charts in Sections 4.4.1 and 4.4.2. All figures are generated by the author.

4.4.1 Graphical presentation of synthetic data set results

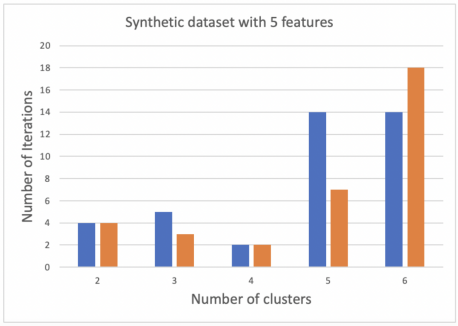


(a) Synthetic data set with 5 features

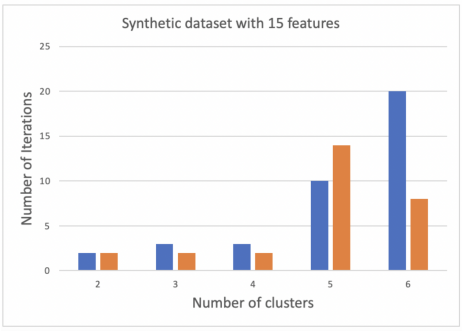


(b) Synthetic data set with 15 features

Figure 11: Silhouette score of synthetic data set

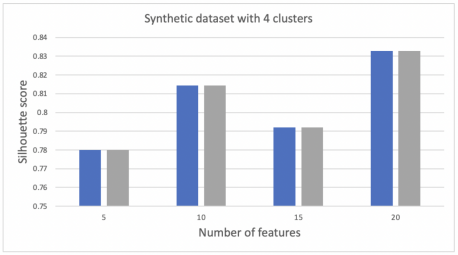


(a) Synthetic data set with 5 features

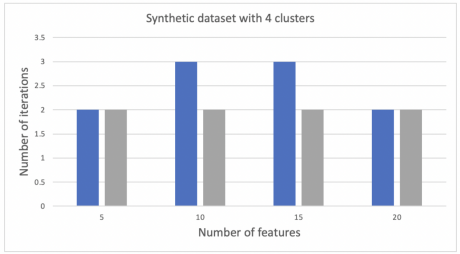


(b) Synthetic data set with 15 features

Figure 12: Number of iterations of synthetic data set



(a) Silhouette score of synthetic data set with 4 clusters



(b) Number of iterations of synthetic data set with 4 clusters

Figure 13: Altering the number of features of synthetic data set with 4 clusters

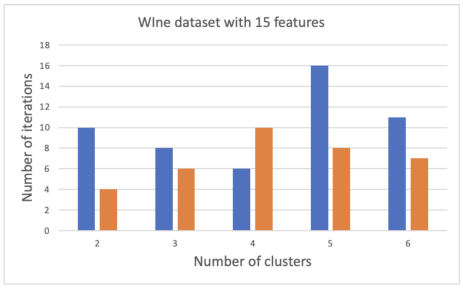
4.4.2 Graphical presentation of wine data set results

(a) Wine data set with 5 features

(b) Wine data set with 15 features

Figure 14: Silhouette score of wine data set

(a) Wine data set with 5 features



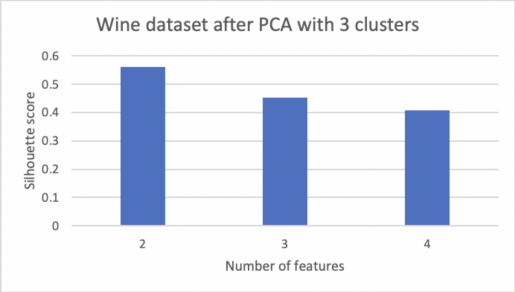
(b) Wine data set with 15 features

Figure 15: Number of iterations of wine data set

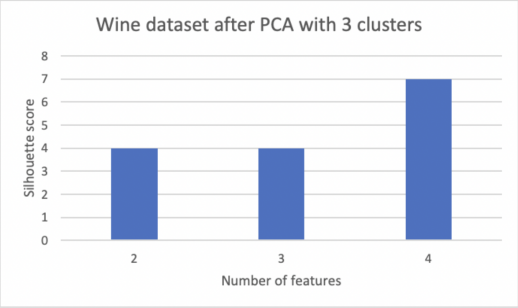
(a) Silhouette score of wine data set with 3 clusters

(b) Number of iterations of wine data set with 3 clusters

Figure 16: Altering the number of features of wine data set with 3 clusters



(a) Silhouette score of wine data set with 3 clusters after undergoing PCA



(b) Number of iterations of wine data set with 3 clusters after undergoing PCA

Figure 17: Wine data set results for 3 clusters after applying PCA

5 Data Analysis

5.1 Analyzing Number of Clusters Using Silhouette Score

First, this investigation has demonstrated that the silhouette score is a reliable indicator of choosing the optimal number of clusters for both synthetic data and real data. In supervised learning, there is a Ground Truth that the algorithms are aware of, meaning that accuracy can be measured. However, k-means clustering is an unsupervised learning algorithm that is learning as it runs with no Ground Truth to compare to, thus accuracy is much harder to measure. For the sake of testing, it was already known that the optimal number of clusters for the synthetic data set was 4. The results shown in Figure 11 portray this, with a peak in silhouette score at 4 clusters for both random and k-means++ initial placements, whether it was data collected for 5 features or 15 features of the synthetic data set. This means that the silhouette score consistently classified 4 clusters as the optimal amount. Although the algorithm is not aware that this is the correct answer, it was already known externally, so this serves as evidence supporting the strength of using silhouette score as an indicator of accuracy.

For the real data collected with the wine data set, it was externally known that the optimal number of clusters should be 3. To see whether the silhouette score was able to capture this optimal cluster value, refer to Figure 14. Although Figure 14a of the wine

data set with 5 features shows a clear peak in silhouette score for 3 clusters, Figure 14b with 15 features shows that the best silhouette score is achieved with 2 clusters. This is a solid example of how real data does not operate like synthetic data, where the optimal number of clusters is the same across all number of features used; instead, it is measured relatively.

5.2 Analyzing Initialization Methods

As seen across all results, the initial placement (random or k-means++) has a negligible difference in results of silhouette score, but causes varying results for the number of iterations. This is especially true for a larger number of clusters when referring to Figure 12, as the number of iterations stays relatively similar regardless of initial placement on both graphs of synthetic data sets with 1-3 clusters, but clusters 4 and 5 see a drastic difference with the left bar representing random initial placement and the right bar representing k-means++ initial placement. However, this pattern is not reflected when looking at data comparing the number of iterations of a synthetic data set with 4 clusters and varying features, as shown in Figure 13b. Instead, a difference is seen between initial placements for the middle two number of features of the data set (10 and 15), with 5 and 20 features achieving the same number of iterations for both initial placements represented by the blue and grey bar graphs. For the wine data set results, as seen in Figures 15 and 16, the initial placements have distinctly different results across all variables tested.

Regardless of what pattern is seen in the differing results between initialization methods, all data collected across both the synthetic and wine data sets reflected that silhouette score was not different between experiments using different initial placements, but the number of iterations always varied at some point.

Since the initialization method does cause varying results for the number of iterations,

its impact is to reduce the number of iterations. Across nearly all data, k-means++ generally outperforms random initialization with a lesser number of iterations used. This is especially when the number of clusters is closer to the Ground Truth known by the external experimenter. This saves computing power and a significant amount of time, especially in large scale data sets such as the wine data set.

5.3 Analyzing the Number of Features

5.3.1 Importance of Feature Scaling

First, it notable that feature scaling in clustering, as different features are being compared. Otherwise, the result has the potential to be severely skewed. In this study, features were already scaled for the synthetic data, but had to be scaled for the wine data. The features within the wine data set included features like the percentage of alcohol and alkalinity of ash, which are two features that cannot directly be compared. Once scaled, the experiment proceeded with the k-means clustering process.

5.3.2 No Direct Relationship Between Features and Clusters

A very significant result is that the number of features has the no direct relationship with finding the optimal number of clusters. It is quite a common assumption that adding more features (including more information) will aid with clustering processes. However, this is not necessarily the case. In Figure 11a and 11b, a varying number of clusters for the synthetic data set with 5 features and 15 features was compared. As seen in the nearly identical results, the number of features varying between 5 and 15 did not create a significant impact in determining the optimal number of clusters. There are slightly more visible varied results between Figure 12a and 12b, comparing a synthetic data set with 5 features and 15 features, but the general pattern is still generally similar. When referring to Figure 13 depicting the silhouette scores and number of iterations of the synthetic data set with a varying number of features, there is no specific pattern seen.

When looking at the wine data set, the effect of the number of features sometimes varies from that of the synthetic data set. As seen in Figure 14a and 14b, the wine data set with 5 features deemed 3 clusters the optimal, but the data set with 15 features deemed 2 clusters the optimal. As previously mentioned, it was already known that the true optimal number of clusters was 3 for the wine data set, which means that the Figure 14a with 5 features was more effective at determining the true number of clusters than Figure 14b with 15 features. When comparing Figure 15a to 15b and Figure 16a to 16b, there was no specific pattern seen when altering just the number of features with a set number of clusters.

Therefore, simply adding more features does not necessarily improve the accuracy of clusters generated by k-means clustering.

5.3.3 Analysis of Dimensionality Reduction Using PCA

Dimensionality reduction using PCA was implemented to reduce the feature dimensionality of the wine data set. The results are shown in Figure 17, the highest silhouette score belongs to 2 clusters, which also has lowest number of iterations when compared with 3 and 4 clusters. This experimental value of the optimal number of clusters matches the Ground Truth. Thus, dimensionality reduction is very helpful in k-means clustering, as this result was not consistently shown in the wine data set prior to dimensionality reduction.

6 Limitations

The first limitation of this study is that only a certain number of clusters and features are considered. Hence, these results can only be considered a local optimal, but not generalized to be the global optimal. Future research needs to be done in order to confirm or deny whether the local results reflect globally.

This investigation only had a Ground Truth to compare experimental results to due to setting up the data set externally for experimental purposes. However, since k-means clustering is unsupervised, no one knows what the number of clusters is - this means multiple must be tested (this experiment tested 5 clusters). Future research can be done to find a Machine Learning approach to figuring out how many clusters should be tested using these methods in order to find the optimal number of clusters to use.

7 Conclusion

In this paper, the effects of changing the number of clusters, number of features, and initialization methods of k-means clustering were analyzed. Logical and mathematical explanations for the patterns observed were also provided.

The results prove that silhouette score is a reliable indicator of accuracy, as there was a Ground Truth to compare experimental results to. However, when k-means clustering is usually run, the Ground Truth is unknown as it is an unsupervised learning algorithm. Since there is a limitation that it is unknown how many clusters should be tested, researchers currently need to test multiple clusters experimentally (such as in this paper) to find the optimal. The amount of clusters and which clusters should be tested can be estimated based on the application of the algorithm. If the ultimate goal is to cluster students into different socioeconomic groups in a high school, it is likely to deduce from logical reasoning that the optimal number of clusters lies between 3 and 5, so a researcher should test the clusters within and around this range (i.e., test 2-6 clusters).

It was found that altering the initialization method had little effect on the silhouette scores, but using k-means++ generally improved computational running speed with a lower number of iterations needed to determine the optimal number of clusters.

The effect of altering the number of features is less predictable, as it followed no clear

relationship for both data sets. However, when the features underwent dimensionality reduction using principal Component Analysis, it was advantageous to improving accuracy and speed with a higher silhouette score and lower number of iterations.

Hopefully this paper will prove useful to Machine Learning resources in guiding their choices as they utilize k-means clustering, leading to more innovative training of unsupervised learning algorithms to be used through all facets of study.

References

[1] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 3 edition, 2011.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[3] Supervised vs. unsupervised learning: What's the difference? `https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning`, 12 March 2021. [accessed 23 June 2022].

[4] Sanjoy Dasgupta and Yoav Freund. Random projection trees for vector quantization. *IEEE Transactions on Information Theory*, 55(7):3229–3242, 2009.

[5] Pulkit Sharma. The most comprehensive guide to k-means clustering you'll ever need. `https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/`, 19 August 2019. [accessed 21 June 2022].

[6] Jake VanderPlas. *Python Data Science Handbook*. O'Reilly Media, Inc., 2016.

[7] A step-by-step explanation of principal component analysis (pca). `https://builtin.com/data-science/step-step-explanation-principal-component-analysis`, 1 April 2021. [accessed 30 June 2022].

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Selecting the number of clusters with silhouette analysis on kmeans clustering.

[10] Michele Forina, Riccardo Leardi, Armanino C, and Sergio Lanteri. *PARVUS: An Extendable Package of Programs for Data Exploration*. 01 1998.

[11] Kmeans silhouette score explained with python example. `https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam`, 17 September 2020. [accessed 14 July 2022].

Appendix

The following program was used for this investigation. Different test trials of k-means clustering algorithms were collected with silhouette score and number of iterations as results. Some insight needed to write this code was drawn from Scikit-learn [9].

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
import time

# Generating the sample data from make_blobs
X, y = make_blobs(
    n_samples=1000,
    n_features= 20,
    centers=4,
    cluster_std=1,
    center_box=(-10.0, 10.0),
    shuffle=True,
    random_state=1,
)
# For reproducibility

# for wine

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine
from sklearn.pipeline import make_pipeline
```

```
30
31
32 features, target = load_wine(return_X_y=True,as_frame=True)
33 # newfeatures = features.iloc[:,0:13]
34 # scaler = StandardScaler()
35 # # transform data
36 # X = scaler.fit_transform(newfeatures)
37
38 # for dimenstion reduction discussion
39 pca = make_pipeline(StandardScaler(), PCA(n_components=4))
40 X = pca.fit_transform(features)
41
42 # print(X)
43
44 range_n_clusters = [2, 3, 4, 5, 6]
45
46 for n_clusters in range_n_clusters:
47     # Create a subplot with 1 row and 2 columns
48     fig, (ax1, ax2) = plt.subplots(1, 2)
49     fig.set_size_inches(18, 7)
50
51     # The 1st subplot is the silhouette plot
52     # The silhouette coefficient can range from -1, 1 but in this
    example all
53     # lie within [-0.1, 1]
54     ax1.set_xlim([-0.1, 1])
55     # The (n_clusters+1)*10 is for inserting blank space between
    silhouette
56     # plots of individual clusters, to demarcate them clearly.
57     ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])
58
59     # Initialize the clusterer with n_clusters value and a random
    generator
60     # seed of 10 for reproducibility.
```

```
61      clusterer = KMeans(n_clusters=n_clusters, init = "k-means++",
    random_state=10)
62      # random_state means that I set a random seed
63      t0 = time.time()
64      cluster_labels = clusterer.fit_predict(X)
65
66      t_batch = time.time() - t0
67      # The silhouette_score gives the average value for all the samples.
68      # This gives a perspective into the density and separation of the
    formed
69      # clusters
70      silhouette_avg = silhouette_score(X, cluster_labels)
71      print(
72          "For n_clusters =",
73          n_clusters,
74          "The average silhouette_score is :",
75          silhouette_avg,
76         # "Time used =",
77         # t_batch,
78          "Kmeans actual iterations =",
79          clusterer.n_iter_,
80      )
81
82      # Compute the silhouette scores for each sample
83      sample_silhouette_values = silhouette_samples(X, cluster_labels)
84
85      y_lower = 10
86      for i in range(n_clusters):
87          # Aggregate the silhouette scores for samples belonging to
88          # cluster i, and sort them
89          ith_cluster_silhouette_values = sample_silhouette_values[
    cluster_labels == i]
90
91          ith_cluster_silhouette_values.sort()
```

```
92
93          size_cluster_i = ith_cluster_silhouette_values.shape[0]
94          y_upper = y_lower + size_cluster_i
95
96          color = cm.nipy_spectral(float(i) / n_clusters)
97          ax1.fill_betweenx(
98              np.arange(y_lower, y_upper),
99              0,
100             ith_cluster_silhouette_values,
101             facecolor=color,
102             edgecolor=color,
103             alpha=0.7,
104         )
105
106         # Label the silhouette plots with their cluster numbers at the
    middle
107         ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
108
109         # Compute the new y_lower for next plot
110         y_lower = y_upper + 10  # 10 for the 0 samples
111
112     ax1.set_title("The silhouette plot for the various clusters.")
113     ax1.set_xlabel("The silhouette coefficient values")
114     ax1.set_ylabel("Cluster label")
115
116     # The vertical line for average silhouette score of all the values
117     ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
118
119     ax1.set_yticks([])  # Clear the yaxis labels / ticks
120     ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
121
122     # 2nd Plot showing the actual clusters formed
123     colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
124     ax2.scatter(
```

```
        X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors,
    edgecolor="k"
    )

    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(
        centers[:, 0],
        centers[:, 1],
        marker="o",
        c="white",
        alpha=1,
        s=200,
        edgecolor="k",
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50,
    edgecolor="k")

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(
        "Silhouette analysis for KMeans clustering on sample data with
    n_clusters = %d"
        % n_clusters,
        fontsize=14,
        fontweight="bold",
    )
    fig.savefig('figures/pca4' + str(n_clusters) + '.png')

```

```
156 plt.show()
```