Università degli Studi di Padova

Department of Ingegneria dell'Informazione

Master Thesis in Ingegneria Informatica

On the use of Silhouette for cost based clustering

Supervisor
Dott. Fabio Vandin
Università di Padova

Master Candidate
Marco Sansoni

Abstract

Clustering plays a fundamental role in Machine Learning. With clustering we refer to the problem of finding coherent groups in a dataset of elements. There are several algorithms to perform clustering that have been proposed in the literature, considering different costs for the optimization problems they consider. In this thesis we study the problem of clustering when the cost function is the silhouette coefficient, an index traditionally used for the internal validation of the results oof clustering algorithms. In particular, we propose and analyze some heuristic algorithms to identify the clustering maximizing the silhouette value, and compare their results with the results from some widely used clustering algorithms.

Contents

Abstract v

List of figures ix

List of tables xi

1 Introduction 1

2 Background 3
- 2.1 Point and Distances 3
- 2.2 Euclidean Distance 4
- 2.3 Jaccard Distance 5
- 2.4 Edit Distance 6
- 2.5 Hamming Distance 6
- 2.6 Clustering 7
- 2.7 K-Means 9
  - 2.7.1 K-Means ++ 11
- 2.8 K-Medoid 13
- 2.9 Clustering Validation 15

3 Related Work 21
- 3.1 An Immune Network Algorithm for Optimization 21
- 3.2 Evolutionary Algorithm for Clustering 22
- 3.3 Soft-Silhouette 23

4 Goal 25

5 Algorithm for Clustering using Silhouette as cost 27
- 5.1 Brute Force Approach 27
  - 5.1.1 Description of the Approach 27
  - 5.1.2 Code 28
  - 5.1.3 Complexity and Correctness 29
- 5.2 Extending K-Means 35
  - 5.2.1 Description of the Approach 35
  - 5.2.2 Pseudo Code 35
  - 5.2.3 Choice of Parameter 40

5.2.4 Complexity and Correctness 41
5.2.5 Improvement 45
5.3 PAMSILHOUETTE 48
5.3.1 Description of the Approach 48
5.3.2 Initial Seeding Technique for Partitioning Around Medoid Algorithm: BUILD 49
5.3.3 Pseudo Code 52
5.3.4 Choice of Parameters 54
5.3.5 Complexity and Correctness 55
5.4 ExtendePAM 58
5.4.1 Description of the Approach 58
5.4.2 Pseudo Code 58
5.4.3 Complexity and Correctness 60
5.5 Improved version of extendePAM, extendePAMv2 63
5.5.1 Complexity and Correctness 64

6 Experimental evaluation 67
6.1 Dataset 67
6.2 Implementation 71
6.3 Comparison of commonly used Clustering Algorithms 72
6.4 Brute Force Approach 75
6.5 Extende K-Means and Improved version 77
6.5.1 Reduced Dataset 77
6.5.2 Complete Dataset 80
6.6 PAMSILHOUETTE 83
6.6.1 Reduced Dataset 84
6.6.2 Complete Dataset 84
6.7 extendePAM 85
6.7.1 Reduced Dataset 85
6.7.2 Complete Dataset 87

7 Conclusion 91

References 94

Listing of figures

2.1 Jaccard Distance is $5/8$. Taken from [1]. 5
2.2 Dendogram representing Hierarchical Clustering. Taken from [1]. 9
2.3 Bad initial seeding. Taken from [2]. 11
2.4 K-medoids versus K-means. Taken from [3]. 14
2.5 Elbow Method. Taken from [4]. 17

6.1 Iris Dataset. Each plot shows distribution of the sample filtering for only 2 features. Taken from [5]. 68
6.2 Reduced Iris dataset. 1,2 and 3 is used to labelling three classes of iris 69
6.3 Reduced Breast Cancer Wisconsin Data Set. 2 and 4 is used to labelling two kinds of cancer, benign or malignant 70
6.4 Reduced banknote authentication dataset. 0 and 1 is used to labelling two different classes of banknote 71
6.5 Histogram of Average Silhouette value on Iris Dataset from most famous clustering algorithm 73
6.6 Histogram of maximum Silhouette value on Iris Dataset from most famous clustering algorithm 74
6.7 Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Iris Dataset 76
6.8 Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Breast Cancer Dataset 78
6.9 Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Banknote Dataset 79
6.10 Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Iris Dataset 81
6.11 Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Breast Cancer Dataset 82
6.12 Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Breast Cancer Dataset 83

7.1 Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Iris Dataset 91
7.2 Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Breast Cancer Dataset 92
7.3 Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Banknote Dataset 93

Listing of tables

6.1 Comparison of Average Silhouette value on Iris Dataset from most famous clustering algorithm . . . 73
6.2 Comparison of maximum Silhouette value on Iris Dataset from most famous clustering algorithm . . . 74
6.3 Silhouette Results performed from BruteForce algorithm on reduced version of Dataset . . . 76
6.4 Evaluation of time required and number of k-subsets for each number of cluster performed on Reduced version of Iris Dataset . . . 77
6.5 Evaluation of silhouette value obtained after execution of $extendeKmeans$ and its improved version on reduced Iris Dataset . . . 77
6.6 Evaluation of silhouette value obtained after execution of $extendeKmeans$ and its improved version on reduced Breast Cancer Dataset . . . 79
6.7 Evaluation of silhouette value obtained after execution of extendeKMeans and its improved version on reduced Banknote Dataset . . . 80
6.8 Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Iris Dataset . . . 80
6.9 Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Breast Cancer Dataset . . . 82
6.10 Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Banknote Dataset . . . 83
6.11 Silhouette value obtained after application of $PAMSILHOUETTE$ algorithm on reduced version of Dataset . . . 84
6.12 Silhouette value obtained after application of $PAMSILHOUETTE$ algorithm on complete version of Dataset . . . 85
6.13 Time in seconds required for application of $PAMSILHOUETTE$ algorithm on complete version of Dataset . . . 85
6.14 Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Iris Dataset . . . 86
6.15 Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Breast Cancer Dataset . . . 87
6.16 Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Banknote Dataset . . . 87
6.17 Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Iris Dataset . . . 88

6.18 Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Breast Cancer Dataset 88
6.19 Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Banknote Dataset 89

1 Introduction

One of the biggest challenges in Machine learning and Data Mining is the detection, classification and validation of cluster. Although there is no single consensus on the definition of a cluster, the clustering procedure can be characterised as the organisation of data into a finite set of categories by abstracting their underlying structure, grouping object in a single partition to describe data according to a similarities or relationship among its object. Clustering could find an usage for several applications, such as identification of customer profile, anomaly detection and market segmentation. A difficulty common to many clustering techniques is that they may need to provide $a - priori$ number of k different clusters in the database. This constraint makes algorithms less powerful and efficient, because they require a domain knowledge for tuning the parameter that fits the data. The lack of a common definition of cluster makes it a widely studied subject, with the presence in literature of many different approaches to the clustering problem. Each technique is characterised by a different metric to allocate the data into relative clusters, each one producing different results. Even the application of the same method may generate, as output, different clusters for each execution of the algorithm. This poses difficulties to users, who not only have to select the clustering algorithm best suited for a particular task, but also have to properly tune its parameters. Such choices are related to clustering validation, another widely researched topics in clustering literature. A common approach to evaluate a quality of a cluster is by the usage of internal criteria. These measures provide a metric of clustering inspecting only the intrinsic information available in the data. Many different criteria have been proposed in literature, however the majority of them are based on the idea of computing the ratio of within-cluster scattering (compactness) to between cluster separation[6]. In this thesis it will be presented some new algorithms for the clustering problem. First of all, we provide a brief overview of the background knowledge required for a later analysis of the content available in the thesis. We highlight the main approach used nowadays for clustering, inspecting some critical aspects for each of them. Furthermore, methods of cluster validation are presented. We refer to them as a tool to evaluate the goodness of a clustering algorithm. Even if it is possible to detect many of them employed into a real scenario, we will focus on the details of a specific statistics, Silhouette. According to its definition we propose some implementations of a clustering technique that aims to its maximisation. For the developing of this specific task we will present several approaches, described into chapter 5. Each algorithm is evaluated analysing pro and cons, with an emphasis on the complexity of the algorithm. In chapter 6 we describe all the experimental results obtained on the application of previous algorithms on 3 dataset, with different number of samples and features. We conclude this thesis with a brief discussion regarding the effectiveness of the algorithm presented compared with the most used clustering algorithm.

2 Background

In the current section we are going to define the basics of clustering, starting with the definition of distance between points, to conclude with the main approach of clustering and commonly used measure for validation.

2.1 Point and Distances

Domain of any clustering problem is a set of finite points X. Considering the common case of the Euclidean space we define a point as $x_i \in X$. Each one has associated a vector of real values, which components are commonly called *coordinates* of the represented points. Besides the definition of a point it is necessary to provide a definition of distance, in fact it can be employed to represent a similarity or dissimilarity between points.
Given a set of point X and $x, y \in X$, a distance measure between them is represented by $d(x, y)$. It has to produce a number as output and it follows the following axioms:

- $d(x, y) > 0$: distance cannot be negative.
- $d(x, y) = 0$ if and only if $x = y$: distance is zero if and only if we consider distance from a point to itself.
- $d(x, z) > d(x, y) + d(y, z)$, with $z \in X$: distance measure must follow triangle inequality.

All the following distance measures need to accomplish the previous property. We provide a short explanation of the main measure distance suitable for an Euclidean Space. It is also relevant to show some other possible distance measure for non Euclidean Space. The following sections mention about the most significant and each of them are suitable for a specific purposes.

2.2 Euclidean Distance

The most known distance measure is the most intuitively and simple approach. We assume n-dimensional points, represented by a vector $x = [x_1, x_2, \ldots, x_n]$. Euclidean distance, also refer as L_2-norm, is defined as:

$$d(x, y) = d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

Euclidean distance is computed summing up the square root of difference in each dimension, and finally take the positive square root. It is necessary to show that the requirements for a distance are satisfied. In fact, the L_2-norm is always positive, as result of positive square root. Then, $(x_i - y_i)^2$ is positive if we can get an occurrence $x_i \neq y_i$ that force the sum to be strictly positive. On the other hand, only if $x_i = y_i \; \forall i$ produces a distance equal to 0.
Last axiom requires to deal with some maths for the proof. However, we can notice that it is a well-known property of Euclidean space. It reminds that the length of the third side of a triangle cannot be longer of the sum of the other two sides.
From the definition of the L_2-norm, we can define the L_r-norm on Euclidean Space, generalizing (2.1) for any constant r. It is defined by:

$$d(x, y) = d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r} \tag{2.2}$$

The case $r = 2$ is the usual L_2-norm mentioned above. Another relevant distance measure is when $r = 1$, it is called L_1-norm or *Manhattan Distance*. It is a measure of the distance that we get travelling from a point to another following grid lines, as the streets of Manhattan. Another important distance is the limit to infinity of 2.2, when r approaches to $+\infty$. It has been called L_∞-norm. Effectively, it can be computed as maximum value of $|x_i - y_i| \; \forall i$.

2.3 Jaccard Distance

Jaccard Distance is a measure useful when we have to deal with sets. In fact, given two sets x and y, it is defined by:

$$d(x,y) = 1 - \frac{x \cap y}{x \cup y} \tag{2.3}$$

Figure 2.1 shows us a possible instance of the sets, for a better comprehension through visualisation. We must now verify the requirements for each distance function.

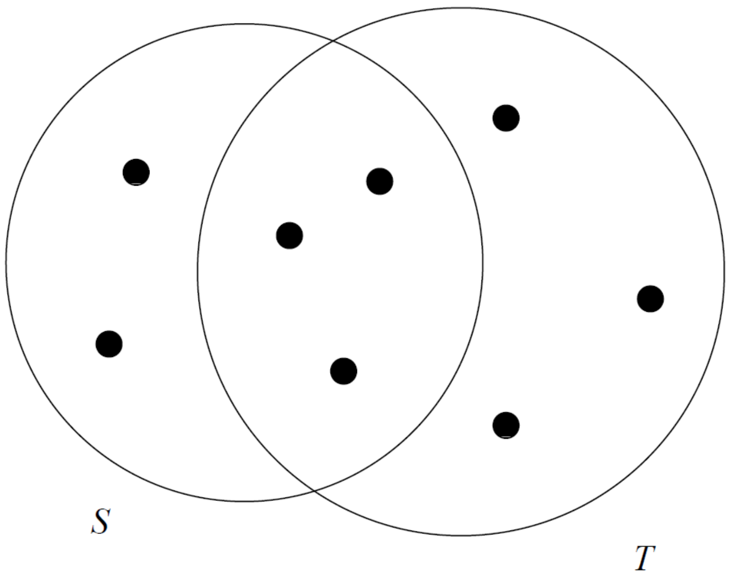

Figure 2.1: Jaccard Distance is $5/8$. Taken from [1].

1. Distance is non negative since the intersection of two sets cannot be bigger than the union.
2. $d(x,y) = 0$ if and only if $x = y$, because to achieve this result we get an intersection equal to the union of two sets, available only when $x = y$. Hence, $x \cup y = x \cap y = x = y$. On the other hand, if $x \neq y$, intersection will always be strictly lower than union.
3. As in the previous section, related to Euclidean distance, proof of triangle inequality requires to deal with a lot of maths. In this thesis the verification will be omitted since it is not relevant for the further developing. In literature there are many proofs of the

triangle inequality related to Jaccard Distance. For instance, a concise proof can be found in [1].

2.4 Edit Distance

This distance is defined for strings. Given two strings $x = x_1x_2 \ldots x_n$ and $y = y_1y_2 \ldots y_n$, an informal definition of this measure is the number of insertions or deletion of single characters that will convert x into y.
For instance, Edit distance between $x = abcde$ and $y = acfdeg$ is 3, since we need to:

1. Delete b.
2. Insert f after c.
3. Insert g after e.

Another way for the computing of this distance is by the usage of *Longest Common Substring(LCS)* of x and y. Once we perform the LCS we can define the Edit Distance as:

$$d(x, y) = |x| + |y| - 2LCS(x, y) \tag{2.4}$$

Surely no Edit Distance can be negative and only two identical strings can have Edit Distance equal to 0. An idea of the proof of the triangle inequality applied to Edit Distance is to note that for converting a string s into a string t we need to turn first into u, and then from u to t. Hence, number of operations done to turn s into t cannot be less than the sum made to convert s in u and finally u in t.

2.5 Hamming Distance

Given a space of vectors, the Hamming Distance between two vectors is the number of components in which they differ. It has a wide appliance related to distance between vector of Boolean, composed only by 1's and 0's. For the sake of completeness, an example is provided. Given $v_1 = 10001$ and $v_2 = 10111$, counting from the right we get that the second and third occurrence are different. So, the Hamming Distance is 2.
The first two requirements are immediately verified, in fact the distance cannot be negative

and it can be equal to 0 only if the vector has the same occurrences, hence the vectors are identical.
As all the others distance measure, Hamming Distance requires more maths, but it should be obvious that it satisfies the triangle inequality. If the distance between a vector x and z are of m components, and z and y differ in n components, it cannot be that x and y differ in more than $m + n$ components.

2.6 Clustering

Recalling the introductory section, a cluster is an organisation of data into a finite set of categories. Since the definition has not a clear and unique value, different clustering algorithms can produce a different division of the points. Hence, we can exploit several approaches of clustering, and we classify algorithms into two main categories, that follow different strategies.

1. Hierarchical or agglomerative
   Algorithms start with each point into a specific cluster. Then, according to the distance measure employed, two cluster "close" are fused into a new cluster. This process is iterated over and over until a convergence criteria stops the algorithm.
   Criteria that can force the conclusion of the algorithm can be several, for instance if we achieve a number of cluster selected a priori, or if the distance between nearest cluster is over a certain threshold. We can also refuse to merge different clusters that lead to point of a single cluster spread in a large region. Once we define the distance measure in the space and stopping criteria S, algorithm can be briefly described by Algorithm 2.1.

   Into Algorithm 2.1 we describe the common algorithm for all the hierarchical clustering approaches. It is executed until S forces termination of the algorithm. Examples of S could be if minimum distance between cluster decreases below a given threshold or if we reach the desired number of clusters. At each iteration of the while loop, two clusters are elected to be merged. Decision of which cluster merge is up to specific linkage criteria of algorithm chosen. Lines 8 and 9 mean that the sample belonging to the cluster to merge are inserted into a new cluster C_l, common for both samples, and previous clusters are deleted.
   Finally, we can describe the process done by the algorithm with the tree in Figure 1.2 showing the complete grouping of points.
   For the proper functioning of the algorithm is important to define also linkage function between two clusters, in order to detect the elected ones to be merged. According to the choice, result may vary. Linkage are defined according to proximity measure among clusters[7]. Principal approaches are:

Algorithm 2.1 Hierarchical Clustering

```
1: Input
2:     $X$    dataset
3:     $S$    stopping criteria chosen for hierarchical clustering
4: Output
5:     $C$    clustering
6: procedure HIERARCHICALCLUSTERING($X$, $S$ )
7:     while $S$ does not stop the algorithm
8:         pick the best two cluster $C_i, C_j$ to merge
9:         $C_l \leftarrow C_i \cup C_j$
10:        $C_i, C_j \leftarrow \emptyset$
11: return $C$
```

- Single Linkage
  Proximity between two clusters is the proximity between their two closest objects. Single linkage method controls only nearest neighbours similarity.
- Complete Linkage
  Proximity between two clusters is the proximity between their two most distant objects.
- Average Linkage
  Proximity between two clusters is the arithmetic mean of all the proximities between the objects of one, on one side, and the objects of the other, on the other side
- Ward Linkage
  In order to define this criteria it is required the definition of within cluster variance. According with [8], the within cluster variance W of a cluster C_k is:

  $$W(C_k) = \sum_{x_i \in C_k} d(x_i, \mu_k) \tag{2.5}$$

  where μ_k is referred to the mean of the cluster C_k, also called centroid. Proximity between two clusters is the within cluster variance of the merged cluster. It aims to minimizes the variance of the cluster being merged.

2. The other class of algorithm involves point assignment.
   Points are considered in some order, and each one is assigned to a cluster with best fit. The algorithm is started with the estimation of initial choice of clusters. In this class of algorithms it is fundamental a domain knowledge of the problem since it is

mandatory the definition of k clusters where the points will be classified. Points are associated to a specific cluster according to an objective function characteristic of the algorithm. For example, points will be associated to the cluster that minimize the L_2-norm between each point and the centre of the cluster which it is classified, as in the K-means algorithm.

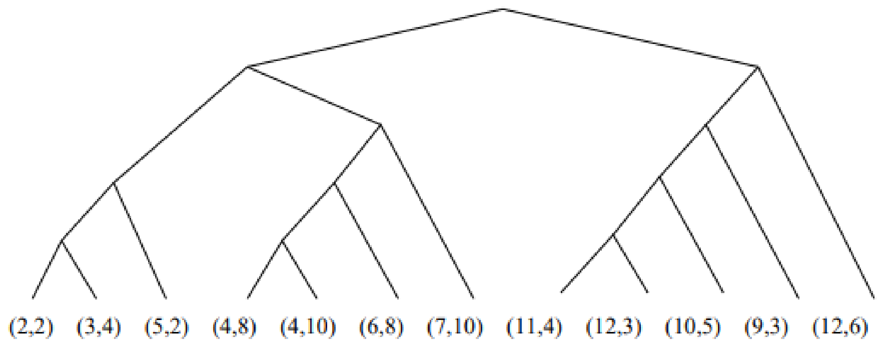

Figure 2.2: Dendogram representing Hierarchical Clustering. Taken from [1].

2.7 K-Means

In this section we shortly explain one of the most famous algorithms of clustering[9]. We initially describe the objective function to be minimized, and then we provide an algorithm that can be really implemented.
They assume Euclidean Space, and the number of cluster k is known in advance.
Given a set of observation $x_1, x_2, \ldots, x_n$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the observation into $k(<= n)$ sets $S = \{S_1, S_2, \ldots, S_k\}$ in order to minimize the within-cluster sum of squares. The purpose is to find:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{2.6}$$

where μ_i is the mean of the point in S_i. Algorithm 2.2 shows an approach of the min problem presented into cost function 2.6. Discussion of the algorithm will be made through inspection of the code shown into Algorithm 2.2
K-means represents just an heuristic solution for the minimization of objective function. In fact, a rigorous solution is computationally expensive, so we look for an approximation of

the optimal result. For a comprehension of the approach we introduce centroid definition. Given a set of d-dimensional points $W = \{w_1, w_2, \dots, w_n\}$, centroid is expressed as:

$$c = \sum_{i=1}^{n} w_i \tag{2.7}$$

Centroid is a d-dimension point $\notin W$, that represents the average value of the points into W.

Algorithm 2.2 K-Means Algorithm

```
1: Input
2:     $X$    dataset
3:     $k$    number of clusters
4: Output
5:     $C$    clustering of dataset into k clusters
6: procedure K-MEANS($X, k$)
7:     $c \leftarrow k$ points that are likely to be in different clusters
8:     make these points centroid of clusters, named $C$
9:     repeat
10:        for all $x_i \in X$
11:            $c_i \leftarrow$centroid which $x_i$ is closest
12:            add $x_i$ to cluster $C_i$
13:        adjust centroid of clusters $C_i$ after addition of $x_i$
14:    until  there are any variations of centroid
15: return C
```

Algorithm 2.2 requires as input k, the number of cluster to partition the given dataset. First line generates c, set of k samples which we assume as cluster center. Selection is a wide argument that we will discuss later. While we denoted with c the set of centroid, we indicate with C, the clustering of the whole dataset, where c_i is the centroid of C_i. Into the while block, at each iteration we evaluate for each sample x_i the centroid closer to it, and we associate each point to closest cluster. Each iteration of the algorithms also include the re calculation of the centroid c, as expressed into 2.7. Termination criteria of the while block is expressed through stability of the centroid. If there are any variations of centroids among two consecutive iteration, algorithm terminates, otherwise it repeats with a new iteration. A crucial operations into the algorithm is the detection of the initial cluster center. Different approaches are presented for the first center selection, a trivial solution is to pick up k points

randomly on the dataset, but more accurate strategies are available in order to achieve better classification. The most used solution for center selection is reported into section 2.7.1.
Key of the algorithm is the repetition of the point assignment. Each point is assigned to the center which minimize its square distance to the cluster, according to:

$$\sqrt{\sum_{i=1}^{d} d(x_i - y_i)^2} \tag{2.8}$$

Point assignment strategy for the clustering suffers of the a priori knowledge of the number of final cluster k. The choice of the best value that fit the point is crucial for a good clustering. To overcome this problem, in the literature are present many approaches of clustering validation, that will be presented in section 2.9. Furthermore, K-means suffers the initial choice of the cluster center. For instance, in the situation displayed in Figure 2.3. Here the example (a)

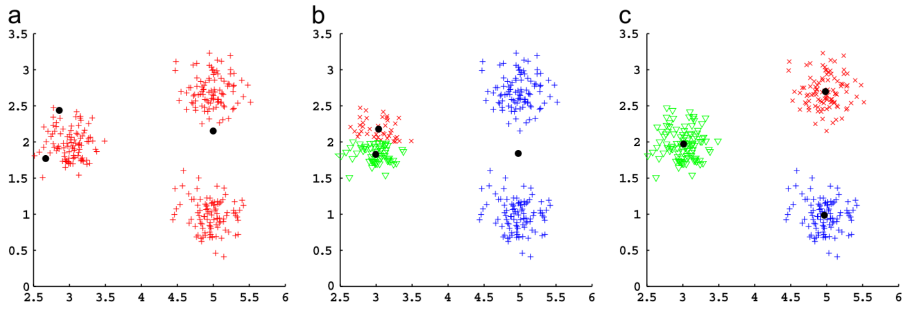

Figure 2.3: Bad initial seeding. Taken from [2].

of a bad initialization that (b) leads to a poor k-Means solution, consisting of clusters with significantly different variances. Instead, a proper seeding leads to example (c). In order to avoid the problem, different techniques are used to reduce probability of such situation.
A trivial solution is to execute more times the algorithm, obtaining different results due to the randomness of the initial choice of the center. Hence, we pick up as the best result the clustering that mostly shows off.

2.7.1 K-Means ++

As explained before, $K - Means$ algorithm begins with k arbitrary centers, typically chosen uniformly at random from the data point. Each point is then assigned to the nearest center, referring to 2.8. At each iteration each center is recomputed as the centroid of all points belonging to it. In this scenario, David Arthur and Sergei Vassilvitskii proposed $K-Means++$ in 2007[10]. It is focused on the appliance of $a-priori$ algorithm focused only into center selection, in order to substantially reduce the probability of the situation occurred in Figure 2.3. It produces centroids used as initial seeding of $K-Means$. Given a set $X=\{x_1,x_2,\dots,x_n\}$, we want to detect k initial centers, namely $C=\{c_1,c_2,\dots,c_k\}$. In particular, let $D(x)$ denote the shortest distance from a data point to the closest center. Finally, algorithm is defined into Algorithm 2.3.

Algorithm 2.3 K-Means ++

```
1: Input
2:     $X$    dataset
3:     $k$    number of clusters
4: Output
5:     $c$    k initial point belonging to dataset used for first center into K-Means
6: procedure K-MEANS ++($X, k$)
7:     $c_1$ ←chosen uniformly at random from $X$
8:     while $|c| < k$
9:         Take new center $c_i$, choosing from $x \in X - C$ with probability $\frac{D(x)^2}{\sum_{x\in X} D(x)^2}$
10: return $c$
```

Algorithm 2.3 is used to center selection, for this reason the number of clusters is a required parameter. Into the algorithm we indicate with c set of centers. First center c_1 is chosen at random from dataset, all the other c_i are chosen according to a probability measure. At the end of while block, c contains exactly k points, used later for first centers into $K-Means$ algorithm.
We can notice that $K-Means++$ is a probabilistic algorithm, it means that the output can differ at each execution. As shown in the Algorithm 2.3, the probability of a point to be chosen as a center is:

$$\frac{D(x)^2}{\sum_{x\in X} D(x)^2} \tag{2.9}$$

Recalling that $D(x)$ is the distance between a point x and its closest center in c, the algorithm highlights that the farthest point are chosen as centers with more probability. This seeding method yields considerable improvement in the final error of $k-Means$. Although the initial selection in the algorithm takes extra time, $k-Means$ converges faster with this initial seeding, lowering the computation time. The $k-means++$ approach has been applied since its initial proposal for center selection.

2.8 K-Medoid

$K-Medoids$ clustering is an approach that aims to minimize a given cost function[11]. Rather than using conventional centroid, it uses medoids to represent the clusters.
Given a set of points $X = \{x_1, x_2, \dots, x_n\}$, a medoids, named x_{medoid} is:

$$x_{medoid} = \underset{yin\{x_1,x_2,\dots,x_m\}}{\arg\min} \sum_{i=1}^{n} d(y, x_i) \tag{2.10}$$

Here we represent with $d(x,y)$ the dissimilarity function between points. It usually refers at the distance L_2 among points. Unlike centroids that usually do not belong to dataset, a medoid is always a member of the data set. It represents the most centrally located item of the data set. The purpose is to determine the best clustering C which minimizes cost function expressed into 2.11. Each clustering C_i contains a point $c_i \in C_i$ which represents a central sample of the cluster.

$$C = \underset{C}{\arg\min} \sum_{j=1}^{k} \sum_{i \in C_j} d(x_i, c_j) \tag{2.11}$$

As for the exact solution of cost function defined into 2.6, even the computation of best clustering C minimizing 2.11 is a NP-Hard problem. An heuristic solution is presented into $PartitionaroundMedoids$ (PAM) algorithm. PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search. Given in input the set of points $X = \{x_1, x_2, \dots, x_n\}$ and the number of cluster k, algorithm can be found into Algorithm 2.4.

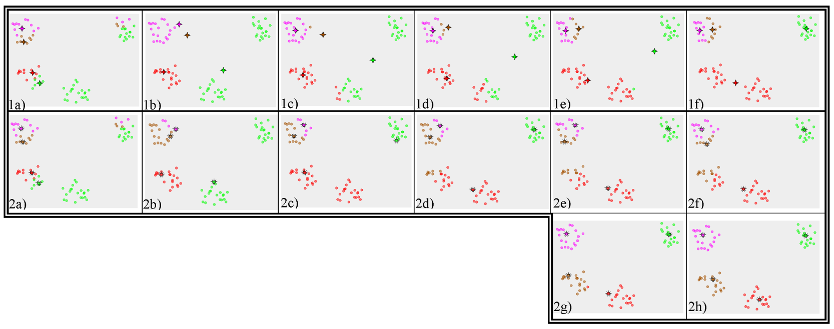

Figure 2.4: K-medoids versus K-means. Taken from [3].

Algorithm 2.4 Partition Around Medoid

```
1: Input
2:     $X$    dataset
3:     $k$    number of clusters
4: Output
5:     $C$    clustering
6: procedure PAM($X$,$k$)
7:     select $k$ of the $n$ data points as the medoids $M$
8:     associate each data point to the closest medoid
9:     while cost of the configuration decreases
10:        for all $m \in M$
11:            for all $o \notin M$
12:                swap $m$ and $o$
13:                associate each data point to the closest medoid
14:                recompute the cost
15:                if total cost increased in the previous step
16:                    undo the swap
       return clustering
```

$PartitioningaroundMedoid$ is described into Algorithm 2.4. As $k - Means$, it requires dataset X, and number of clusters as input of the algorithm. Initially it selects k points as medoid, each point into M has to belong to the dataset. All the other samples are associated to the closest medoid. For each configuration of the set M, cost is the sum of distance from each sample to the closest medoid. Core of the algorithm is the continue swap among a point $m \in M$ and $o \notin M$. At each iteration is performed a change among m and o, defining a new sets of medoid. If the new cost function associated to the set of medoids increases, swap is undone. If no swap achieves a reduction of the cost, algorithm concludes reporting final clustering, ones with smallest cost function.

These algorithms suffer of the same issues of $k-Means$ algorithm, since they are heuristic solutions, they cannot achieve certainly an exact solution, but an approximate result. There are several different strategies for picking up the initial medoid during initialization phase of the algorithm. As for $k - Means$, a simple solution involves selection of medoid uniformly at random from the data point. Most sophisticated approaches can be found in literature to achieve the best solutions at the first executions.
As well as $k - Means$ it requires as input the number of cluster k. Figure 2.4 highlights a dataset which PAM algorithm performs better than $k - Means$, subdividing samples into the obvious cluster structure of data set. The image 1a-1f presents a typical example of the k-means convergence to a local minimum. In this example, k-medoids algorithm, showed into images 2a - 2h, with the same initial seeding of k-means converges to the obvious cluster structure.
K-medoids method is more robust than k-means in presence of noise and outliers because a medoids is less influenced by outliers or other extreme values[12].

2.9 Clustering Validation

Since there is not a unique definition of clusters, and some different strategies are proposed, we need to identify a score to classify the results provided from algorithms. In order to overcome this issue a lot of strategies have been produced.
Generally, clustering validation statistics can be categorized into 3 classes:

1. Internal Cluster Validation
   It uses internal information of the clustering process to evaluate the goodness of a clustering structure without reference to external information. It can be used for estimating the number of clusters and the appropriate clustering algorithm without any external data.

2. External Cluster Validation
   It consists in comparing the results of a cluster analysis to an externally known result, such as externally provided class labels. It measures the extent to which cluster labels match externally supplied class labels. Since we know the “true” cluster number in advance, this approach is mainly used for selecting the right clustering algorithm for a specific data set.

3. Relative Cluster Validation
   It evaluates the clustering structure by varying different parameters for the same algorithm, such as the number of clusters k. It is generally used for determining the optimal number of clusters.

In this section we will focus on exploiting the intrinsic structure of the data, discussing about internal cluster validation. Here, we present some metrics available in literature.

Elbow Method

The Elbow Method [13] looks at the percentage of variance explained as a function of the number of clusters: one should choose a number of clusters so that adding another cluster does not give much better modelling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the ”elbow criterion”. This ”elbow” cannot always be unambiguously identified.
In Figure 2.5 we indicate with a red circle the elbow of the plot. According to this method, the proper number of cluster for such algorithm is 4.
This is one of the most naive approaches. Other similar approaches are available inspecting the average diameter of the cluster instead of the variance. However, a common aspect is identifying k according to the ”elbow” of the plot. The main issue of this approach, as several other techniques of cluster validation, is the fact that it is mandatory to perform clustering with all the possible value of k, in order to build the graph.

Dunn Index

Dunn Index is a metrics used to evaluate clustering algorithm. It was introduced by J.C. Dunn in 1974[14] [15]. The aims are to identify sets of clusters that are compact, with a small variance between members of the cluster. For a given assignment of cluster, high value of Dunn index means better clustering of data. One of the drawbacks is the computational cost required for the computation of the index.
As a requirement for comprehension of Dunn Index we provide a digression on the size or diameter of a cluster. There are different approaches, as the mean distance between two point in a cluster, but for this statistic we consider the farthest two points inside a cluster.
Let C_i a cluster of vectors, and define a distance function d. Let x and y be two m-dimensional

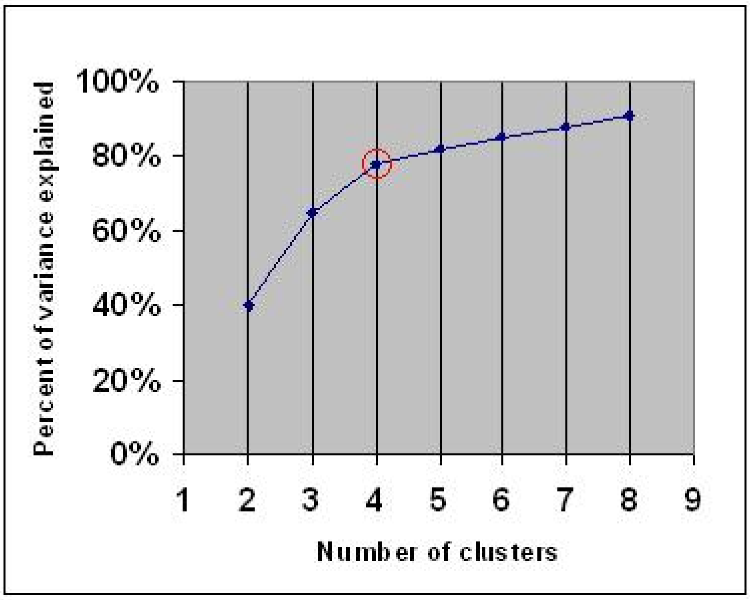

Figure 2.5: Elbow Method. Taken from [4].

vector belonging into the same cluster C_i. The diameter of the cluster is:

$$\Delta_i = \max_{x,y \in C_i} d(x, y) \tag{2.12}$$

Besides the definition of the diameter of the cluster, we require a measure of inter cluster distance metric. Hence, we define:

$$\delta(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \tag{2.13}$$

While distance intra cluster is computed referring to the farthest two points, we consider nearest points for the inter cluster measure. We can finally provide a definition of the Dunn statistic. If we get k cluster, Dunn Index for the set is defined as:

$$DI_k = \frac{\min_{1 \leq i \leq j \leq k} \delta(C_i, C_j)}{\max_{1 \leq i \leq k} \Delta_i} \tag{2.14}$$

This formulation has a peculiar problem, in that if one of the clusters is badly behaved, where the others are tightly packed, since the denominator contains a 'max' term instead of an average term, the Dunn Index for that set of clusters will be uncharacteristically low.

Silhouette Coefficient

Silhouette coefficient has been introduced for the first time by Peter J. Rousseeuw in 1987 [16].
It refers to a method of interpretation and validation of consistency within clusters of data. The silhouette value is measured of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). A formal definition of the two previous aspects will be required for a later analysis.
For cohesion we mean how closely are the points into a given cluster. Optimal cluster will be characterized from a high value of cohesion. For the Silhouette coefficient we require to compute, for each point i, the statistics related to the distance from i to all other points in its own cluster, in order to compute the average distance.

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in Ci, j \neq i} d(j, i) \tag{2.15}$$

where we consider that $i \in C_i$, and $|C_i|$ as the cardinality of the cluster.
Cluster separation is a statistic that evaluate how distinct or well-separated a cluster is from other clusters. For Silhouette analysis is required to compute the distance between a given point i and any other cluster, which i is not a member. First, we need to perform the average dissimilarities between i and the other cluster, computed as the average distance between the point and all the members of that cluster.

$$d(i, C_j) = \frac{1}{|C_j|} \sum_{j \in C_j} d(i, j) \tag{2.16}$$

The equation 2.15 is performed for each cluster $C_j \neq C_i$ obtained from the clustering algorithm. Once, we highlight a "neighbouring cluster", and the statics b_i used in silhouette is the average distance between that specific cluster. Hence, we get:

$$b_i = \min_{j \neq i} d(i, C_j) \tag{2.17}$$

With the previous metrics we can define silhouette index, for a given point i as:

$$s_i = \frac{b(i) - a(i)}{\max\{a_i, b_i\}} \tag{2.18}$$

Which can be also written as:

$$s_i = \begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases} \tag{2.19}$$

From this definition is clear that $-1 \leq s_i \leq 1$.

Also, note that score is 0 for clusters with size = 1. This constraint is added to prevent the number of clusters from increasing significantly.

For s_i close to 1 we require $a_i << b_i$. As a_i is a measure of how dissimilar i is to its own cluster, a small value means it is well matched. Furthermore, a large b_i implies that i is badly matched to its neighbouring cluster. Thus an s_i close to one means that the data is appropriately clustered. If s_i is close to negative one, then by the same logic we see that i would be more appropriate if it was clustered in its neighbouring cluster. An s_i near zero means that the sample is on the border of two natural clusters.
Silhouette index is calculated separately for each point. In order to provide a metric useful to determine the quality of the clustering it is necessary to combine all the indexes.
The average s_i over all points of a cluster is a measure of how tightly grouped all the points in the cluster are. Thus the average s_i over all data of the entire dataset is a measure of how appropriately the data have been clustered. It can be used to determine the natural number of a cluster into a dataset, computing the index for each possible k in order to select the maximum.
Our thesis will be focused on this particular index, providing a clustering technique aimed to maximizing the silhouette, as we reported into chapter 4, explaining the goal of this thesis.

3 Related Work

One of the main issues into clustering problem is related to the asymptotic complexity. In fact, finding an optimal solution to the partition of N object into k cluster is NP-Complete and, provided that the number of distinct partitions of N data into k clusters increases approximately as $k^N/k!$, attempting to find a globally optimal solution is usually not feasible. This difficulty has stimulated the development of efficient approximated algorithms. Genetic algorithms are widely believed to be effective on NP-complete global optimization problems and they can provide good sub optimal solutions in reasonable time. Under this perspective in literature are available different algorithms that are based on genetics. In this scenario we present two approaches found in literature. Last section discusses about a possible extension for Silhouette metrics, in order to decrease its asymptotic complexity.

3.1 An Immune Network Algorithm for Optimization

A model based on immune networks, is the $aiNet$ algorithm [17]. This algorithm was successfully applied to several problems in data compression and clustering. An optimization version of the $aiNet$ algorithm, called $opt - aiNet$, was developed with the ability to perform unimodal and multimodal searches [18].
In $opt - aiNet$ each network cell represents a solution to the problem being treated. The algorithm is used to find solutions to continuous optimization problems, in which each cell is a real valued vector in an Euclidean space. The $opt - aiNet$ algorithm uses an evalua-

tion function to be optimized, which also provides the fitness value of that cell. The fitness measures the quality of the solution represented by a cell: high fitness values indicate good solutions, while low fitness values indicate lower quality solutions. There are also the clone generation and mutation processes: at each generation there is only cell proliferation of a number of clones defined by the user, and in the mutation there is the variation of these clones, based on a specific criterion. In $optaiNet$ the mutation rate is proportional to the fitness: cells with high fitness values suffer low mutation rates, while cells with low fitness suffer high mutation rates.
In addition to the fitness of a cell, which measures its quality in relation to a cost function to be optimized, a cell also has an affinity, representing how similar it is to other cells in the network. The affinity of a cell may lead to a cell cloning or pruning. The affinity of cells is evaluated based on the Euclidean distance among them. Cells with an affinity greater than a pre-defined threshold may be pruned. After this suppression process new randomly generated cells are included in the network.

3.2 Evolutionary Algorithm for Clustering

The evolutionary algorithms have their inspiration in the Darwinian Theory of Evolution. These algorithms are able to find good solutions to complex problems in reasonable computational time. The $EvolutionaryAlgorithmforClustering$ (EAC) was proposed to achieve optimal groupings of data [19].
The EAC is started by a population of individuals, randomly generated, which represent candidate solutions to the data clustering problem. This initial generation is then used to produce offspring through preselected random variations. The resulting candidate solutions will be evaluated based on their effectiveness in solving the problem.
As the same way that the environment makes a selective pressure on individuals, in the evolutionary algorithm there is also a process where only the most adapted individuals (best fitness) are preserved to the next generation, and this process is repeated several times[19]. In each iteration they also provide an evaluation of the clustering based on the silhouette functions. The stopping criterion in the EAC is the maximum number of iterations (num_it) that is parameterized; this criterion is usually present in evolutionary algorithms.
.

3.3 Soft-Silhouette

Since it was created, Silhouette has become one of the most popular internal measures for clustering validity evaluation. In [20, 21] it is compared with a set of other internal measures and proven to be one of the most effective and generally applicable measures. However, when Silhouette is applied in the evaluation of $k - Means$ clustering validity, many more extra calculations are required, and the extra calculations increase following a power law corresponding to the size of the dataset, because the calculation of the Silhouette index is based on the full pairwise distance matrix over all data. This is a challenging disadvantage of Silhouette. From this perspective, Silhouette needs to be simplified for $k - Means$ to improve its efficiency.
Simplified Silhouette was, to our knowledge, first introduced by Hruschka in [19], and used as one of the internal measures in his following research. It inherits most characteristics from Silhouette and therefore can be used in the evaluation of the validity of not only a full clustering but also a single cluster or a single data point. On the other hand, the distance of a data point to a cluster in Simplified Silhouette is represented with the distance to the cluster centroid instead of the average distance to all (other) data points in the cluster, just as in the $k - Means$ Cost Function.
Recalling definition for Silhouette statistics, we can define as well simplified Silhouette. In 2.15, given a point i, we define its function a_i as the average distance between all the other points in the same cluster. Similarly, we define a'_i as the distance between the selected point and the centroid of the cluster:

$$a'_i = d(i, C_i) \tag{3.1}$$

where we assume that $i \in C_i$. We need to provide a new approach also for the function b_i, defined into 2.17, so we get:

$$b'_i = \min_{i \neq h} d(i, C_h) \tag{3.2}$$

Once we provide the below definition, Silhouette is computed as usual as:

$$ss_i = \frac{b'_i - a'_i}{max\{a'_i, b'_i\}} \tag{3.3}$$

Simplified Silhouette index is performed as the average value of 3.3 computed for each point of the data set. Furthermore, to prove the effectiveness of such value, we provide a short theoretical comparison between the original statistics and the simplified one.

The overall complexity of the computation of Silhouette is estimated as $0(n^2)$, while that of Simplified Silhouette is estimated as $O(kn)$. [19]. A detailed discussion of the asymptotic complexity of Silhouette is provided into section 5.1.3.
When k is much smaller than n, Silhouette is much more computationally expensive than Simplified Silhouette. In addition, during the process of $k - means$ clustering, the distance of each data point to its cluster centroid has already been calculated in each iteration, which greatly reduces the calculation of both the Cost Function and Simplified Silhouette. Therefore, the Cost Function and Simplified Silhouette are much more efficient in the evaluation of $k - means$ clustering validity.
Once we have highlighted the benefits gained from the usage of the Simplified Silhouette, we can achieve in literature some review done with this statistics. In [22] we found a comparison between the metrics provided by Silhouette and its Simplified version as a tool for clustering validation. The article highlights that the result from two metrics are comparable in different test cases.

4
Goal

The objective of this thesis is the development of an algorithm which, given a dataset of point as input, it produces a clustering as output. We focus on generation of cluster which maximize the Silhouette index. This metrics is usually used to provide an evaluation related to the quality of a cluster, as a validation criteria. Into this thesis we focus on a development of a clustering which aims to produce clustering with maximum value of Silhouette, regardless the number of clusters. Unfortunately, computation of an exact algorithm for pursuit of the objective requires an exponential time, since it fails within NP-Hard problem. The purpose of this thesis is achieved with an heuristic solution which aims a best possible approximation of the real result.
Many famous clustering algorithms provide heuristic solution for cost based clustering. For instance, cost functions 2.6 and 2.11 are respectively related to $k - Means$ and $k - Median$ algorithm. We provide here a cost function based on the silhouette index defined in 2.18. Formally, if a dataset $X = \{x_1, \ldots, x_n\}$ of n samples is given, the cost function aims to detect k clusters $C = \{C_1, \ldots, C_k\}$ which maximize the following cost function:

$$\operatorname*{argmax}_{C} \sum_{i=1}^{n} \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.1}$$

In this scenario we need to maximize a cost function, while, the other algorithm aims to minimize the value. It is due to Silhouette index, which higher value means better clustering.

Some proposal will be developed and discussed in terms of time required for the termination of the algorithm and silhouette performance. Furthermore, for each algorithm we evaluate its asymptotic complexity.
It is important to highlight that cost function defined in 4.1 does not involve any restriction on k, the number of clusters required. This constraint is found into algorithms like $k-Means$ and $PartitioningAroundMedoid$, even if their cost function does not require such value. In this thesis, we will provide some implementations which aim to maximize 4.1 without requiring further constraints.

5
Algorithm for Clustering using Silhouette as cost

In the following chapter we are focusing on some possible solutions to the goal described in the chapter above. We provide here some different ways to achieve the purpose. For each implementation proposed we describe the algorithm, through usage of the pseudo code. For each algorithm we provide all the theoretical knowledge useful to a full understating. Furthermore, it will be crucial also for the asymptotic analysis supplied into each approach. Each algorithm will be accompanied with a proof of termination, to highlight the effectiveness of the proposal.

5.1 Brute Force Approach

5.1.1 Description of the Approach

The first implementation of the problem it is the simplest possible solution, brute force approach.
In order to discover the best possible clustering solution that maximize silhouette index, we generate all the possible clustering of the given dataset. For each one we evaluate Silhouette index in order to store as result clustering which maximizes Silhouette. The advantage of this approach is the lack of any constraint related to the number of clusters as output, in fact the algorithm evaluates all the possible partitions. Furthermore, once the algorithm partitions

input data set and test each combination we have the guarantee of finding the best solution as claimed by the goal section. Moreover termination of brute force is guarantee. This implementation is provided also as a reference for the further algorithms described in this thesis. In fact, we retrieve from this implementation the best possible value of the silhouette metrics that we can achieve from a given dataset. On the other hand, since this approach has to generate all the possible subset, cost function of the algorithm is exponential. A detailed measurement of the asymptotic complexity of the algorithm is provided into section 5.1.3.

5.1.2 Code

The code reported into this section is useful for a better comprehension of the solution and it is useful also for analysis of complexity.

Algorithm 5.1 Brute Force Algorithm

```
Input
    $X$    dataset
Output
    $Y$    clustering with best value of silhouette
procedure BRUTEFORCE(X)
    $s \leftarrow -1$
    $Y \leftarrow \emptyset$
    for $k \leftarrow 2\ to\ |X|$
        $X_a \leftarrow \emptyset$
        $Y_a \leftarrow \emptyset$
        $a_k \leftarrow algorithm\_u(X, k)$
        for all $a \in a_k$
            for $c_{k,a} \leftarrow 1\ to\ |a|$
                for $e_{k,a,c} \leftarrow 1\ to\ |a^{c_{k,a}}|$
                    $X_a \leftarrow X_a \cup e_{k,a,c}$
                    $Y_a \leftarrow Y_a \cup c_{k,a}$
            $s_a \leftarrow silhouette(X_a, Y_a)$
            if $s_a > s$
                $s \leftarrow s_a$
                $Y \leftarrow Y_a$
    return $Y$
```

Algorithm 5.1 reports pseudo code useful to describe the brute force approach. It is required dataset X as input, without any restrictions on the number of clusters. s is referred to the initial value of Silhouette, stored initially at -1, worst possible condition. Y instead contains the best possible clustering, initialized with empty set. For each value of k, X_a and Y_a contains the temporary value of dataset and its related clustering. a_k stores all the possible $k - subset$ obtained from the dataset, for the given k. Loop, available from line 12 up to line 16, stores for each combination of the $k - subset$ sample into X_a and relative cluster into Y_a. Finally, silhouette of the combination is stored into s_a, and best silhouette value and best clustering is updated if $s_a > s$. It returns best clustering achieved after inspection of all the possible k, from 2 up to $n - 1$.

5.1.3 Complexity and Correctness

For a better analysis of this implementation we need a digression related on how we can generate all the possible partitions from a given dataset X. In the algorithm task is performed from $algorithm\_u$ procedure. It generates all the k-subset of given X.
A k-subset of set X is a partition of all the elements in X into k non-empty subsets. For instance, given a set $X = 1, 2, 3$, a 2-subset returns all the possible way to split into 2 subsets the elements of X. Therefore we get

$$1|23 \quad 2|13 \quad 3|12 \tag{5.1}$$

which vertical line is used in order to split one subset to the other one. In order to find all the possible partitions of the set X, we need to perform the algorithm for each value $k \leftarrow 1, |X|$. With the previous set X, there are in total 5 partitions:

$$123 \quad 1|23 \quad 2|13 \quad 3|12 \quad 1|2|3 \tag{5.2}$$

In this list the element in each subset can be written in any order, because $13|2$, $31|2$, $2|13$ and $2|31$ represent the same partition. We can standardize the representation listing all the elements of each subset in increasing order, and arranging the subset in increasing order of their smallest element. In this way, all the partition returned from $X = 1, 2, 3, 4$ are:

$$\begin{array}{c} 1234 \quad 123|4 \quad 124|3 \quad 12|34 \quad 12|3|4 \quad 134|2 \quad 13|24 \\ 13|2|4 \quad 14|23 \quad 1|234 \quad 14|2|3 \quad 1|24|3 \quad 1|2|34 \quad 1|2|3|4 \end{array} \tag{5.3}$$

According to [23] one of the most convenient ways to represent a set partition inside a computer is to encode it as a restricted growth string, a string $a_1a_2\dots a_n$ which:

$$a_1 = 0 \textit{ and } a_{j+1} \leq 1 + max(a_1, \dots, a_j) \quad for \quad 1 \leq j < n \tag{5.4}$$

The idea is to set $a_j = a_k$ if and only if $j = k$, and to choose the smallest available number for a_j whenever j is smallest in its subset. Restricted growth string for 5.3 is:

$$\begin{array}{cccccccc} 0000 & 0001 & 0010 & 0011 & 0012 & 0100 & 0101 & 0102 \\ 0110 & 0111 & 0112 & 0120 & 0121 & 0122 & 0123 & \end{array} \tag{5.5}$$

The algorithm used to generate all the partition in the pseudo code listed above exploits all the restricted growth string visiting all partitions of X that satisfies condition 5.4. A detailed description of the code can be found here [23].
For later analysis it is important to evaluate the number of set partitions. In the previous we have seen that there are 5 partitions for $X = 1, 2, 3$ and 15 for $X = 1, 2, 3, 4$. A quick way to compute the count was presented here [?], which define the following triangle of numbers.

$$\begin{array}{cccccc} 1 & & & & & \\ 2 & 1 & & & & \\ 5 & 3 & 2 & & & \\ 15 & 10 & 7 & 5 & & \\ 52 & 37 & 27 & 20 & 15 & \\ 203 & 151 & 114 & 87 & 67 & 52 \end{array} \tag{5.6}$$

Here the entries $\omega_{n1}, \omega_{n2}, \dots, \omega_{nn}$ belonging to the $n - th$ row are obtained by:

$$\begin{aligned} \omega_{nk} &= \omega_{(n-1)k} + \omega_{n(k+1)} \textit{ if } 1 \leq k < n \\ \omega_{nn} &= \omega_{(n-1)1} \textit{ if } n > 1 \\ \omega_{nn} &= 1 \textit{ if } n = 1 \end{aligned} \tag{5.7}$$

In this triangle, entry on the diagonal and in the first column are the total number of set partitions, also known as Bell Numbers. We denote that with ω_n. The first cases are:

n	$=0$	1	2	3	4	5	6	7	8	9	10
ω	$=1$	1	2	5	15	52	203	877	4140	21147	115975

The Bell numbers $\omega_n = \omega_{n1}$ for $n \geq 0$ must satisfy the recurrence formula:

$$\omega_{n+1} = \omega_n + \binom{n}{1}\omega_{n-1} + \binom{n}{2}\omega_{n-2} + \cdots = \sum_k \binom{n}{k}\omega_{n-k} \tag{5.8}$$

because each partition of $\{1, \ldots, n+1\}$ is obtained choosing k elements from $\{1, \ldots, n\}$ to store in subset containing $n+1$ and partitioning the remaining elements in ω_{n-k} way, for some k. We can found here [23] a demonstration of an asymptotic estimates of the number of total partition of a given set. We report here the final result:

$$\omega_n \approx O(n/logn)^n \tag{5.9}$$

Here, we made an analysis on the number of the partition produced as output from the algorithm. The last step is the evaluation of complexity in time. Inspecting the code, available here [16], each partition is generated, exploiting restricted growth string, into constant time. Time complexity differs from the space complexity just for a constant value.

Other fundamental aspects on the algorithm is the computation of the silhouette score for each sample. Recalling the definition of Silhouette in 2.18, for each sample $i \in X$, we need to compute the statistics of the dissimilarity a_i in a cluster, and also the statistics of dissimilarity between cluster b_i. A simple algorithm to calculate a_i is showed into algorithm 5.2.

The code follows the definition provided in 2.15. Moreover, the time complexity of the algorithm for the sample $x_i \in C_i$ is $O(|C_i|)$. It depends obviously from the size of the cluster which points belong. In a worst case scenario, where all samples belong to the same cluster, we get a complexity of $O(n)$.
The pseudo code available into Algorithm 5.3 can be used for the computation of the statistics b_i.

In order to evaluate b_i for a given point $x \in C_i$, we need to compute all the distances between x and $x_j (\in C_j)$, where $x \neq x_j$ and $C_i \neq C_j$, in order to detect the minimum. Assuming X partitioned in k clusters, asymptotic time complexity of the algorithm is $O(n-$

Algorithm 5.2 Statistics a_i

```
1: Input
2:     $x$      point belonging to dataset to be classified
3:     $C_i$    cluster where $x \in C_i$
4: Output
5:     $a$      statistics a associated to $x$
6: procedure A($x$,$C_i$)
7:     $s \leftarrow 0$
8:     for all $x_j \in C_i$
9:         if $x \neq x_j$
10:            $s \leftarrow s + d(x, x_j)$
11: return $\frac{s}{|C_i|-1}$
```

Algorithm 5.3 Statistics b_i

```
1: Input
2:     $x$      point belonging to dataset to be classified
3:     $C_i$    cluster where $x \in C_i$
4:     $C$      clustering of the whole dataset
5: Output
6:     $b$      statistics b associated to $x$
7: procedure B($x$,$C_i$,$C$)
8:     for all $C_j \in C$
9:         if $C_j \neq C_i$
10:            $s_j \leftarrow 0$
11:        for all $x_j \in C_j$
12:            $s_j \leftarrow s_j + d(x, x_j)$
13: return $\min_i \frac{s_i}{C_i}$
```

$|C_i|)$ because $n-|C_i|$ operations are required to compute all pairwise distances. It is possible to compute minimum value in an efficient way in $O(1)$ by the usage of a variable which contains the updated minimum.
As the previous statistics, we get worst case if C_i is composed by only the sample x. Hence, asymptotic time complexity is $O(n)$. Recalling silhouette index defined in 2.18, we need to perform both indexes to perform metrics of a single sample. Therefore, the asymptotic complexity of silhouette index for the sample $x_i \in C_i$ is $O(n)$. Silhouette index for the whole clustering is expressed with

$$S = \sum_{i=1}^{n} s_i \tag{5.10}$$

Asymptotic time complexity for the evaluation of the silhouette index for a given set X and clustering $C_i, C_2, \ldots, C_k$ is:

$$O(n^2) \tag{5.11}$$

Previous section provides us all the tools required for a proper analysis of complexity of this implementation. We split the cost function of the entire problem into a sum of cost function of each main operation. Main tasks are the following:

- Generating all the k-subset through $algorithm\_u(X, k)$

  Generation phase of all the possible k-subset of a given set is computed into the $algorithm\_u$ procedure. Assuming that each combination is generated in constant time, we obtain the following cost function:

  $$T_1(n, k) = O(\omega_k) \tag{5.12}$$

  which a proper definition of ω_k is available in 5.7 - 5.9.

- Operations done for each partition It can be useful inspecting the cost required for each partition found, so we need to focus from line 9 up to line 16 of the pseudo code. We obtain that:

  $$T_2(n, k) = \sum_{i=1}^{n} \alpha_1 + n^2 = \alpha_1 n + n^2 \tag{5.13}$$

  In 5.13 we collect all the operation done with the assignment of a cluster at each point x. Silhouette function need to be performed at each iteration, and its asymptotic time complexity is shown in 5.11.

Finally we can define a more concise cost function for the whole algorithm. We get:

$$\begin{aligned} T(n,k) &= \sum_{k=2}^{n}\left(T_1(n,k) + \sum_{i=1}^{|\omega_k|} T_2(n,k)\right) \\ &= \sum_{k=2}^{n}\left(\omega_k + \sum_{i=1}^{\omega_k} \alpha_1 n + n^2\right) \\ &= \sum_{k=2}^{n}\big(\omega_k + (\alpha_1 n + n^2)\omega_k\big) \\ &= \sum_{k=2}^{n}\big(\omega_k(n + n^2 + 1)\big) \\ &= (n^2 + \alpha_1 n + 1)\sum_{k=2}^{n}\omega_k \end{aligned} \tag{5.14}$$

Through the usage of 5.9, we can define asymptotic complexity of the algorithm:

$$O\left(\frac{n^{n+2}}{logn^n}\right) \tag{5.15}$$

Last step of the analysis is related to evaluation of correctness of the algorithm.

Proposition 1. *Given a set of points X. The algorithm $BruteForce$, with X as input always finishes. (1). Furthermore, it provides as output a set Y with $|Y|$. Here $y_i = j \in Y$ means that x_i belongs to cluster C_j. Clustering $C_1, C_2, \ldots, C_k$ gained from Y represents $argmax_{C_1,C_2,\ldots,C_k} \sum_{i \in C_1,\ldots,C_k} \frac{b(i)-a(i)}{max(b(i),a(i)}$ (2)*

Proof. In order to proof point (1), it is necessary to underline how there are any variation on the variables constituting the loop. At each iteration a new partition is analyzed and nested loop from line 8 to line 10 iterates over all the cluster into all partitions. Assignment does not involve any variable, so termination is guarantee. The proof of point (2) is done by contradiction. Initially we assume that it exists a partition a_γ which leads to cluster which maximize our goal, and it is not returned from our algorithm. We know that a_γ is composed by k clusters $C_{\gamma,0}, \ldots, C_{\gamma,k}$. Silhouette metrics needs a number of cluster from 2 to $|X|$, hence k will be into this bound. Since a_γ is the maximum we have that:

$$s_\gamma = \sum_{i \in C_{\gamma,0},\ldots,C_{\gamma,k}} \frac{b(i)-a(i)}{max(b(i),a(i)} > s_j = \sum_{i \in C_{j,0},\ldots,C_{j,k}} \frac{b(i)-a(i)}{max(b(i),a(i)} \forall j \tag{5.16}$$

which $C_{j,0}, \ldots, C_{j,k}$ is the partition generated from the $j - th$ partition returned from $algorithm\_u$ procedure. It means that a_γ is a partition that it is not generated from the $algorithm\_u$ procedure, but it is contradictory since it returns all the k-partition of X.

$\square$

5.2 Extending K-Means

5.2.1 Description of the Approach

According with asymptotic time complexity described into bruteForce algorithm, the main issue of the previous approach is due to the exact computation of the best result. Recalling that clustering is a NP-hard problem, the following implementations aims to approximate the best result through an heuristic solution.
Since the most famous clustering algorithm is $k - Means$, we develop an extension of it. The idea is to execute the algorithm, and then we slightly adjust clustering in order to increase, if possible, silhouette value. This approach involves definition of bound acts to detect samples of dataset which we would like to classify again. It is done through a threshold of the silhouette values.
An issue to overcome is due to the intrinsic constraint into $k - Means$ clustering where the number of clusters is defined as an input of the problem, while purpose of the algorithm is the deletion of this constraint. To overcome this problem algorithm is computed for each k, until a stopping condition is triggered.
In this implementation, if the silhouette for a given k is below a certain threshold, algorithms stops, reporting the results. This solution increases enormously the complexity of the algorithm. This approach is also affected from the intrinsic problem of the $k - Means$ algorithm, where more executions of the same algorithm return different results due to the initial seeding.

5.2.2 Pseudo Code

In this section we describe with pseudo code the algorithm proposed. It is composed by a main program which recall results produced by other procedures.

Algorithm 5.4 Extension of K-Means

```
1:  Input
2:      $X$    dataset
3:  Output
4:      $C_*$    clustering which maximizes Silhoutte index
5:  procedure EXTENDEKMEANS($X$)
6:      $C_* \leftarrow \emptyset$
7:      $S_*, S_{-1} \leftarrow 0$
8:      $k \leftarrow 2$
9:      $\lambda_t \leftarrow \frac{17}{20}$
10:     while $S_{-1} \geq \lambda_t S_*$ $and$ $k < n - 1$
11:         $C_k \leftarrow K - means(X, k)$
12:         $\psi_0, \psi_1 \leftarrow min(silhouette(C_k)), max(silhouette(C_k))$
13:         $\lambda_c \leftarrow \frac{10(\psi_1 - \psi_0)}{logK + 130} + \psi_0$
14:         $M_k \leftarrow generateMPoint(X, C_k, \lambda_c)$
15:         $T_k \leftarrow closestCenter(M_k, C_k, k)$
16:         $S_k, C_k \leftarrow bruteForce(X, T_k, C_k)$
17:         if $S_k > S_*$
18:             $S_* \leftarrow S_k$
19:             $C_* \leftarrow C_k$
20:         $k \leftarrow k + 1$
        return $C_*$
```

Algorithm 5.4 described above is a main function which , during its execution, call several other procedures. It requires as input the dataset of point X, in order to produce the best clustering, reported as output C_*. First algorithm initializes all the different variables, we get C_* which contains the clustering with best value of Silhouette S_*. S_{-1} contains the initial value of silhouette, while λ_t and λ_c are thresholds defined into section 5.2.3. In order to define the threshold λ_c we require the maximum and minimum score of Silhouette computed on C_k, and they are stored in ψ_1 and ψ_1 respectively. Algorithm is repeated until the best Silhouette achieved for the current k, S_k is bigger than a threshold, or we test all the possible k. At each iteration it executes $k - Means$ algorithm with current value of k. Clustering C_k contains the result of $k - means$, and it is used as input of procedure $generateMPoint$. This function returns the set of points M. T_k reports value of the closest two clusters for each point into M, in fact into $closestCenter$ procedure, each point of M_k is associated with one of the two closest centers. Into $bruteForce$ we test each possible combination of the sample T_k. It returns as output S_k and C_k, the best silhouette value achieved after all combination and its relative clustering, respectively. Finally, it updates variable S_* and C_*, which contains the best silhouette value among all the possible S_k.

Algorithm 5.5 generateMPoint

```
1: Input
2:     $X$    dataset
3:     $C$    current clustering of the dataset
4:     $\lambda$    threshold used to select point to store in $M$
5: Output
6:     $M$    set of points belonging to dataset to be reclassified
7: procedure GENERATEMPOINT($X$,$C$,$\lambda$)
8:     $M \leftarrow \emptyset$
9:     $S \leftarrow silhouette(X, C)$
10:    for all $x_i \in X$
11:        if $S_i < \lambda$
12:            $M \leftarrow M \cup x_i$
       return $M$
```

Algorithm 5.5 reports a procedure used from $extendeKMeans$. It executes silhouette score for each sample in X. Set M contains all the point that its silhouette score is below the given threshold λ.

Algorithm 5.6 closestCenter

```
1: Input
2:     $C$    dataset
3:     $M$    set of points belonging to dataset to be reclassified
4:     $C$    clustering of the whole dataset
5: Output
6:     $T$    array containing for each sample into M lables of two closest clusters
7: procedure closestCenter($M$,$C$,$k$)
8:     $T \leftarrow \emptyset$
9:     $CC \leftarrow center(C)$                ▷ $CC_i$ = center's coordinates of i-th cluster
10:    $D^1, D^2 \leftarrow \infty$
11:    $C^1, C^2 \leftarrow -1$
12:    for all $x_j \in M$
13:        for $i \leftarrow 1\ to\ k$
14:            $d_{i,j} \leftarrow distance(x_j, CC_i)$
15:            if $d_{i,j} < D^1$
16:                $C^2, D^2 \leftarrow D^1, C^1$
17:                $C^1, D^1 \leftarrow i, d_{i,j}$
18:            else $d_{i,j} < D^2$ & $d_{i,j} > D^1$
19:                $C^2, D^2 \leftarrow i, d_{i,j}$
20:        $T_j \leftarrow (C^1, C^2)$
21: return $T$
```

$closestCenter$ procedure returns for each point selected in M, index of two closest center, stored into T. Selection of the closest center are performed through distance among the point and the center of the cluster.
First list is initialized empty. CC stores the coordinates of the centroid of each cluster. For instance, CC_i contains coordinates of $i-th$ cluster. For each sample into M, D^1, D^2 store temporary distance to the closest cluster, and to the second closest cluster. Index of related clusters are saved into C^1 and C^2. The core of the algorithm is the loop from line 12 up to lines 14. At each iteration it inspects distance among $x_j \in M$, and all the cluster centers, available into CC. Then, it stores into C^1 and C^2, the index of closest cluster and second closest cluster, respectively. Finally, for each x_j it saves into T_j the values C^1 and C^2. As

notation we refer as the $j-th$ point into M with T_j. Its closest cluster is $T_j[0]$, while the second closest is $T_j[1]$.

Algorithm 5.7 bruteForce

```
1: Input
2:     $C$    dataset
3:     $T$    array containing for each sample into M lables of two closest clusters
4:     $C$    clustering of the whole dataset
5: Output
6:     $S_*$   best silhouette obtained
7:     $C_*$   clustering related to best silhouette
8: procedure BRUTEFORCE($X$,$T$,$C$)
9:     $S_* \leftarrow -1$
10:    $C_* \leftarrow \emptyset$
11:    $max \leftarrow 2^{|T|}$
12:    for $i \leftarrow 0$ $to$ $max$
13:        $b \leftarrow binaryRepresentation(i)$
14:        for $j \leftarrow 0$ $to$ $|b|$
15:            $C[T_j] \leftarrow T_j[b_j]$        ▷ $C[T_j]$ is the labels associated to j-th point in T ≡ M
16:        $S_i \leftarrow silhouette(X, C)$
17:        if $S_i > S_*$
18:            $S_* \leftarrow S_i$
19:            $C_* \leftarrow C$
20: return $S_*, C_*$
```

Algorithm 5.7 performs the bruteForce analysis of all the combination. Since for each $x_i \in M$ we get two possible way to classify each point, we exploit binary representation. We would like to test each possible combination of associating x_i the cluster $T_i[0]$ or $T_i[1]$, and test for all i. For a binary representation, if sample X_i belong with $T_i[0]$, it has associated the flag 0, otherwise 1. Iterating over all the possible binary numbers from 0 to $2^{|M|}$ and consider each digit as a flag for the point, we can generate and test all the possible combinations. For instance, assuming as input:

$$M = 1, 2, 3, 4 \tag{5.17}$$

and closest center, stored into T, are:

$$T_i[0] = 0 \quad T_i[1] = 1 \quad for\ i = 1, \dots, 4 \tag{5.18}$$

All the possible combination are obtain through binary representation of the number from 0 to $2^4 = 16$. We have:

$$0000 \quad 0001 \quad \dots \quad 1110 \quad 1111 \tag{5.19}$$

Associating the $i - th$ point into T with the $i - th$ bit of the representation we test all combinations. For example, with 0001, we have

$$C[T_1] = 1 \quad C[T_2] = 0 \quad C[T_3] = 0 \quad C[T_4] = 0 \tag{5.20}$$

Finally, for each combination, it is performed Silhouette index, storing the best value. In this scenario, with the binary combination of 0000, each point into M is associated with its closest center. It is the same clustering generated through $k - Means$. In fact, Algorithm 2.2 highlights that $k - Means$ is stopped when all the $x_i \in X$ are associated to the closest center. A detailed discussion of $k - Means$ algorithm is available into section 5.2.4.

5.2.3 Choice of Parameter

The fundamental idea behind this implementation is to detect points which have a low value of Silhouette and it tries to reclassify into another cluster these samples, in order to increase Silhouette value of the whole clustering. It is crucial for the effectiveness of the algorithm to define a threshold λ_c where a point x_i, with silhouette value below it, needs to be stored into M, waiting to be reclassified. Small value of λ_c involves few points into M and consequently slightly change in term of silhouette, or even it leads to any variations from $k - Means$. On the other hand, huge value of λ_c causes a huge number of samples into M, and an exponential increase of time required for the termination of the algorithm. Before the definition of threshold we need to define two variables.

$$\psi_0 = min(silhouette(C_k)) \tag{5.21}$$

With 5.21 we refer to the minimum score obtained among all the silhouette values computed on clustering C_k. As highlighted in the pseudo code we refer with C_k as the result of the

$k - Means$ algorithm. Analogously we define:

$$\psi_1 = max(silhouette(C_k)) \tag{5.22}$$

where we refer as the maximum value obtained from silhouette among all the samples in C_k.

$$\lambda_c = \frac{10(\psi_1 - \psi_0)}{logK + 130} + \psi_0 \tag{5.23}$$

Another parameter is required for the proper functioning of the algorithm. In fact, $extendeKMeans$ iterates over all the possible k values. As an improvement a stoppable condition is introduced. When the algorithm starts detecting a cluster with a value of Silhouette below threshold it returns the best value achieved at that time.

$$SilhouetteValue(currentK) < \lambda_d maxValueSilhouette \tag{5.24}$$

Even in this case it is crucial the detected value of the bound. In fact, a small value of the threshold involves repetition of the algorithm for large value of k, when Silhouette is sensibly smaller than the best value achieved. On the other hand, a bigger value risks the algorithm to stop if we detect for some k a small reduction of Silhouette, but it ignores the possibility of a subsequently increase of Silhouette, for larger k We prefer a larger bound than a tighter one avoiding an undesired termination. The following bound is defined after these considerations and we get:

$$\lambda_t = \frac{17}{20} \tag{5.25}$$

5.2.4 Complexity and Correctness

Since each iteration of $extendeKMeans$ begins with execution of $k-Means$, it is required a digression on analysis of $k-Means$, for a complete evaluation of the algorithm. Its pseudo code is available into Algorithm 2.2.

First, we evaluate the cost function required for each iteration. It is based on the number of samples n and on the number of clusters k. First k points, as explained into the section 2.7.1, can be produced into different ways. In this analysis we assume that they are picked up randomly, hence time required for generation of the first k center is negligible. Into $T_1(n, k)$

we store cost function required for each iteration of the algorithm.

$$\begin{aligned} T_1(n,k) &= \left(\sum_{i=1}^{n}\left(\sum_{j=1}^{k}\alpha_1\right)+\alpha_2\right)+\sum_{i=1}^{n}\alpha_3 \\ &= \sum_{i=1}^{n}(\alpha_1 k+\alpha_2)+\sum_{i=1}^{n}\alpha_3 \\ &= n(\alpha_1 k+\alpha_2)+\alpha_3 n \end{aligned} \tag{5.26}$$

which $\alpha_1, \alpha_2, \alpha_3$ represent constant factors. Total cost function of the $k-Means$ algorithm is:

$$T(n,k,i) = i\left(n\left(\alpha_1 k+\alpha_2\right)+\alpha_3 n\right) \tag{5.27}$$

which we introduce i in order to denote the number of iterations. The last analysis involves the evaluation of asymptotic complexity of i.
Into a real scenario $k - Means$ performs very well, with a small number of iterations even for large dataset. It is still required the evaluation of the complexity of the algorithm in worst case scenario. Recalling the results highlighted found in [24] we have:

$$i = 2^{\Omega(\sqrt{n})} \tag{5.28}$$

which it highlights an exponential growth of the algorithm. In conclusion, asymptotic time complexity is:

$$O(nk2^{\Omega\sqrt{n}}) \tag{5.29}$$

Through the usage of 5.29, we compute tha asymptotic complexity of $extendeKMeans$ algorithm into worst case scenario.
For a proper discussion of the cost function involved in the algorithm, we split it among procedures recalled into $extendeKMeans$. At the end we sum up the results in order to detect the asymptotic complexity. The common input variable of the cost function is the number of samples n and the number of clusters k.

- generate MPoint
  This procedure produces as output the set of point that they will be reclassified later. Dimension of the output is related of the value of threshold λ_c. A more appropriate analysis will be held later, in order to produce an upper bound of the complexity of the whole algorithm.

Cost function for this procedure is noted as $T_1(n,k)$.

$$T_1(n,k) = \alpha_1 n^2 + \sum_{i=1}^{n} \alpha_2 = \alpha_1 n^2 + \alpha_2 n \tag{5.30}$$

where $\alpha_1 n^2$ represents complexity of Silhouette function, as described in 4.11. α_2 is a constant factor linked to comparison and assignment into the loop.

- closest Center
  The procedure aims to associate to each point into M the reference to the closest center. We denote the cardinality of the set M with $|M|$. Computation of the center of each cluster is performed into $\alpha_1 n$. Final cost function of the procedure is the following:

$$\begin{aligned} T_2(n,k) &= \alpha_1 n + \sum_{i=1}^{|M|} \sum_{j=1}^{k} \alpha_2 \\ &= \alpha_1 n + \alpha_2 k\,|M| \end{aligned} \tag{5.31}$$

- bruteForce
  Final procedure is executed in order to testing all the possible combinations into T, output of the $closestCenter$ procedure. Since $|T| \equiv |M|$, we represent with $|M|$ the cardinality of T. Binary representation of a number is assumed completed into constant time. Cost function is stored into $T_3(n,k)$.

$$\begin{aligned} T_3(n,k) &= \sum_{i=0}^{2^{|M|}} \left(\sum_{i=0}^{|M|} \alpha_1 + \alpha_2 n^2 + \alpha_3 \right) \\ &= \sum_{i=0}^{2^{|M|}} \left(|M|\,\alpha_1 + \alpha_2 n^2 + \alpha_3 \right) \\ &= 2^{|M|} \left(|M|\,\alpha_1 + \alpha_2 n^2 + \alpha_3 \right) \end{aligned} \tag{5.32}$$

  which α_1 is constant factor related to distance computation, $\alpha_2 n^2$ is silhouette index and α_3 is related to final comparison and assignment into the algorithm.

Since all the methods recalled in the main function are exploited, we can analyze the cost function of the entire algorithm. As $k - Means$, the core of the algorithm is the repetition of the analysis for increasing k, until the stopping condition denoted into 5.24 forces the termination. Even if in the real case scenario the result is better on average, asymptotic complexity discussion is referred to the worst case. In this situation algorithm iterates over all the possible k, since $k = n - 1$. For the generation of the threshold λ_c we need to detect the minimum and maximum of silhouette score, cost function for this task is:

$$T_4(n, k) = \alpha_1 n^2 + \alpha_2 n \tag{5.33}$$

where we recall cost function of Silhouette as highlighted in 5.11 and searching for the maximum and minimum value are linear operations executed into an array.
Summing up all the previous result we obtain a cost function $T(n, k)$.

$$\begin{aligned} T(n,k) = & \sum_{i=2}^{n-1} (nk2^{\Omega\sqrt{n}} + \alpha_1 n^2 + \alpha_2 n + \alpha_1 n^2 + \alpha_2 n + \alpha_1 n + \alpha_2 k \left|M\right| + \\ & \qquad + 2^{|M|}(\left|M\right| \alpha_1 + \alpha_2 n^2 + \alpha_3) + \alpha_4) \\ = & (n-1)(nk2^{\Omega\sqrt{n}} + \alpha_1 n^2 + \alpha_2 n + \alpha_1 n^2 + \alpha_2 n + \alpha_1 n + \alpha_2 k \left|M\right| + \\ & \qquad + 2^{|M|}(\left|M\right| \alpha_1 + \alpha_2 n^2 + \alpha_3) + \alpha_4) \end{aligned} \tag{5.34}$$

We need to define the value of $|M|$. Into the worst case scenario the number of points included into M are the whole dataset, if the silhouette is below threshold for each point. Hence in the worst case $M = O(n)$. Asymptotic complexity of the algorithm is:

$$O(n^3 2^n) \tag{5.35}$$

The last part of the analysis involves the correctness of the algorithm.

Proposition 2. *Given a dataset of n points $X = \{x_1, x_2, \dots, x_n\}$, the application of Algorithm 5.4 returns a value of Silhouette S_*, generated from a clustering C_*. $S_* \geq S_k \quad \forall k = 2, \dots, n-1$. Furthermore, Algorithm 5.2 always finishes.*

Proof. First of all we proceed with the proof that algorithm always finishes. Since it is an iterative procedure we need to proof that variables that define the loop do not change during the execution of the inner code, hence the loop reaches its end. It is simple to proof this condition into the procedure $generateMPoint$, $closestCenter$ and $bruteForce$. It requires a small analysis of the condition into while block inside the $extendeKMeans$ function. The termination of the loop occurs when the silhouette starts decreasing with reference to a given threshold. If this condition never occurs, $k < n-1$ guarantees termination since k increases at each iteration regardless of the result of clustering. Hence, the algorithm reaches termination.

For proving $S_* \geq S_k$ we need to proof that exists a combination where the algorithm executes the same clustering of $k - Means$, guaranteeing a value of $S_* = S_k$.

For each point stored into set M, $closestCenter$ detects the two closest clusters to each point, respectively C^1 and C^2. All other points are labelled with the same value returned from $k - Means$. First, we prove that exists a condition where each point is labelled with its closest center. During the execution, $bruteForce$ algorithm tests for each point in M assigning them to closest or second closest cluster. Each test is related to a binary number. The binary sequence $00 \ldots 0$ is referred to the selection for each point of the closest cluster C^1. In $bruteForce$ it is the first sequence analyzed into for loop. $extendeKMeans$ tests this clustering, storing its silhouette Value. Last step of the demonstration involves the proof that into $k - Means$ each point is labelled with the closest center.

We proceed through a contradiction. Suppose that it exists $x_i \in C_i$, with:

$$d(x_i, C_i) > d(x_i, C_j) \textit{ for a given } 0 \leq j \leq k. \tag{5.36}$$

Since $k - Means$ is concluded, in the last iteration, there are no variations of the cluster center, so all the points were perfectly matched with the proper cluster. In the line 12 of Algorithm 2.2 we perform the cluster selection of all samples, matching each point with closest cluster. During the execution of the algorithm, in the last iteration, x_i should be classified into C_j since 5.36 occurs. In this case, the centroid would have been different and another iteration was launched, but it contradicts the hypothesis given. We have proven that it exists a clustering which associate each point with the closest center, as depicted in $k - means$. In conclusion silhouette value returned from Algorithm 5.4 must be:

$$S_* \geq S_k \quad \forall k \tag{5.37}$$

$\square$

5.2.5 Improvement

In the following section we proposed an improved version of the algorithm $extendeKMeans$ discussed above. It is focused on define in a precise way $|M|$, in order to lower the asymptotic complexity of the algorithm from exponential to polynomial. All the procedures described in the section above are still valid, hence we only need to redefine the procedure $generateMPoint$. In this approach we drop the use of the bound λ_c, which defines a threshold useful to deciding whatever a point belong to M. At each iteration of the algorithm cardinality of set M is $log_2 n$. The new procedure $generateMPointv2$ is defined into Algorithm 5.8.

Algorithm 5.8 generateMPointv2

```
1:  Input
2:      $X$   dataset
3:      $C$   current clustering of the dataset
4:  Output
5:      $M$   set of points belonging to dataset to be reclassified
6:  procedure GENERATEMPOINTV2($X$,$C$)
7:      $M \leftarrow \emptyset$
8:      $S \leftarrow silhouette(X, C)$
9:      $S_* \leftarrow sorted(S)$
10:     for $i \leftarrow i\ to\ log_2(n)$
11:         $j \leftarrow \arg\min S_{*,i}$
12:         $M \leftarrow M \cup x_j$
13: return $M$
```

The algortihm 5.8 executed the procedure $generateMPointv2$, applied into Algorithm 5.4 instead of the procedure $generateMPoint$. It returns the set M of points that they need to be reclassified. It gives up previous approach based on threshold, and it defines a fixed number of ”worst” value. It stores into S the silhouette score for each sample, then this list is sorted and stored into S_*. Lines from 11 up to 13, scan the $log_2 n$ points with the worst value of Silhouette storing them into M, the output of the procedure.
There are many possible alternatives to store the first k samples. If the solutions adopt the usage of a sorting algorithm with complexity $O(nlogn)$,it does not affect the asymptotic complexity. The improvement of this solution can be analyzed in terms of asymptotic complexity. In the final cost function in 5.34, we need to replace the value of cost function $T_1(n, k)$,

with the following:

$$\begin{aligned} T_{11}(n,k) &= \alpha_1 n^2 + \alpha_2 nlogn + \sum_{i=1}^{log_2 n} \alpha_3 \\ &= \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n \end{aligned} \tag{5.38}$$

In order to define the cost function of the all algorithm we need to substituting $T_1(n,k)$ into 5.34 with $T_1 1(n,k)$, and remove cost required for the definition of the threshold λ_c. We obtain:

$$\begin{aligned} T(n,k) = & \sum_{i=2}^{n-1} (nk2^{\Omega\sqrt{n}} + \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n + \alpha_1 n + \alpha_2 k\left|M\right| + \\ & + 2^{|M|}\left(\left|M\right|\alpha_1 + \alpha_2 n^2 + \alpha_3\right) + \alpha_4) \\ = & (n-1)(nk2^{\Omega\sqrt{n}} + \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n + \alpha_1 n + \alpha_2 k\left|M\right| + \\ & + 2^{|M|}(\left|M\right|\alpha_1 + \alpha_2 n^2 + \alpha_3) + \alpha_4) \end{aligned} \tag{5.39}$$

Since before, an improvement is achieved, into the worst case, M could be the whole dataset, now we certainly assume that $|M| = log_2 n$. With this substitution into 5.39 we obtain:

$$\begin{aligned} T(n,k) = & (n-1)(nk2^{\Omega\sqrt{n}} + \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n + \alpha_1 n + \alpha_2 k log_2 n + \\ & + n\left(\alpha_1 log_2 n + \alpha_2 n^2 + \alpha_3\right) + \alpha_4) \end{aligned} \tag{5.40}$$

From 5.40 we can derive the asymptotic complexity of the new algorithm, that is:

$$O(n^2 2^{\Omega(\sqrt{n})}) \tag{5.41}$$

It is important also to notice that this factor is due to $k - Means$ procedure and its worst case scenario, that it rarely occurs. The results gained from this algorithm S_* has the same property of results obtained into the first implementation, it means $S_* \geq S_k \quad \forall k$ since proof does not involve size of set M. The termination of this new algorithm is simply provided by analyzing that procedure $generateMPointv2$ always reaches a termination. Proof of correctness of this implementation descends immediately from the previous one.

5.3 PAMSILHOUETTE

In chapter 5.1 we adopted a naive solution for the definition of the best clustering which maximizes silhouette, with the brute force approach.
Later, into section 5.2 we proposed an algorithm which provides an extension of the most famous technique used for clustering, $k - Means$. It runs the algorithm and then it performs an improvement of the Silhouette with slight change to the results produced. Here, we would like to provide another extension from one of the main famous algorithms, $k - Medoid$. Starting from the idea behind that the algorithm we proposed is an improvement suitable to increase performance for the main task of this thesis.

5.3.1 Description of the Approach

In this implementation we would like to refine the choice of the medoids of the Partitioning Around Medoid algorithm in order to maximize the silhouette of the clustering. Once we set the medoid C, each point $X_i \in X$ will be classified in the proper cluster C_i according with:

$$C_i = \underset{c_j \in C_1,\ldots,C_k}{\arg\min}\ d(x_i, c_j) \tag{5.42}$$

While in the PAM algorithm we would like to minimize 2.10, the distance between points and closest medoid, in this implementation target function will be the Silhouette index of the clustering. Given the number of the cluster k, PAM maximizes over all potential medoid M the function:

$$f(M) = -\sum_j d(x_j, M) \tag{5.43}$$

which $d(x_j, M)$ represents dissimilarities between point x_j and medoid M. In the comparison described above it is associates with the euclidean distance (2.1) between x_j and its closest medoid. The proposal of this implementation is the replacement of $d(x_j, M)$ in 5.43 with the silhouette defined in 2.18. This new approach, named $PAMSILHOUETTE$ aims to maximize the following function:

$$f(M) = \sum_j S(j) \tag{5.44}$$

$S(j)$ function is the silhouette index of the cluster returned applying 2.1 to each points in order to associate them to the closest medoid. $PAMSILHOUETTE$ would iterate through the possible combinations of medoids in order to detect which one provides the best result in term of Silhouette. Furthermore, we use an iterative optimization algorithm to detect the best solution, instead of trying all the possible combinations.

5.3.2 Initial Seeding Technique for Partitioning Around Medoid Algorithm: BUILD

One of the main known issue of PAM technique is the high difference in the silhouette from two executions of the algorithm. It is much more sensible than other clustering approaches to the initial seeding.
In literature, as $k - Means++$ introduces a technique for choosing the initial center of the clustering, even PAM has an initialization phase, called $BUILD$ [25], which an initial clustering is obtained through the successive selection of representative object until k objects have been found. Given a set of points $X = \{x_1, x_2, \dots, x_n\}$ and a target number of cluster k, the purpose of the $BUILD$ algorithm is the selection of k points, named $c_1, c_2, \dots, c_k$ as representative for clustering named respectively $C_1, C_2, \dots, C_k$. In the first phase an initial clustering is obtained through the successive selection of the representative objects until k points have been found.
The first medoid c_1 is the one which minimizes the sum of distances to all other objects:

$$c_1 = \arg\min_{1 \le h \le n} \sum_{j=1}^{n} d(x_h, x_j) \tag{5.45}$$

c_1 is the most centrally located point into X. Subsequently, at each step, another object is selected, decreasing the objective function of $k - Medoid$ (2.11) as much as possible. This medoid has a minimal distance to all the selected medoids and the distance to this object is the smallest.

$$c_2 = \arg\min_{1 \le h \le n} \sum_{j=1}^{n} \min(d(x_j, c_1), d(x_j, x_h)) \tag{5.46}$$

$$c_3 = \arg\min_{1 \le h \le n} \sum_{j=1}^{n} \min(\min_{1 \le l \le 2} d(x_j, c_1), d(x_j, x_h))) \tag{5.47}$$

$$\dots$$

$$c_k = \arg\min_{1 \le h \le n} \sum_{j=1}^{n} \min(\min_{1 \le l \le k-1} d(x_j, c_1), d(x_j, x_h))) \tag{5.48}$$

This process is continued until k object has been found [25]. Here is provided the psuedo code suitable for its implementation. M represents the distance matrix among all objects in X, while d_j represents the distance from x_j to the closest medoid.

Algorithm 5.9 BUILD Algorithm

```
1: Input
2:     $M$    matrix of distances
3: Output
4:     $C$    initial seeding for Partitioning Around Medoid
5: procedure BUILD(M)
6:     for $i \leftarrow 1$ $to$ $n$
7:         if $\sum_{j=1}^{n} m_{ij}$ $is$ $minimal$
8:             $c_1 \leftarrow x_i$
9:     $D$ ←distances to the nearest medoid
10:    for $l \leftarrow 2$ $to$ $k$
11:        for $i \leftarrow 1$ $to$ $n$
12:            if $\sum_{j=1}^{n} min(d_j, m_{ij})$ $is$ $minimal$
13:                $c_l \leftarrow x_i$
14:        Update $D$
15: return $C$
```

The algorithm 5.9 executes the selection of the initial medoid as described into 5.45 - 5.48. The input of the algorithm is the matrix of distances among all the sample M. m_{ij} is referred to the distance between x_i and x_j.The first medoid c_1 is selected into the first iteration, from line 6 up to line 8. It implements 5.45.
Others medoids, from c_2 up to c_k, are retrieved into the second for loop, they respects 5.46 - 5.48 and they are stored into C, which it is the output of the procedure.
For a complete treatment of the algorithm we perform the asymptotic complexity in time of $BUILD$. It is crucial to evaluate if the amount of time required into $BUILD$ for the generation of the seeding medoid reflects into a faster execution of PAM.
Here we focus in $BUILD$ algorithm, without consider the time involved for the computation of the distance matrix M required as input for the procedure. On specifics we compute the cost function for three categories which we can subdivide the algorithm

- Detection of the first medoid c_1
  Cost function associated is $T_1(n, k)$. We obtain the value inspecting first loop into the pseudo code. We get:

$$T_1(n,k) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_1 = \alpha_1 n^2 \tag{5.49}$$

  which we assume as constant α_1 the time spent for detection of the minimal sum of distances and assignment to c_1. Since value of m_{ij} is provided from input, it does not depend on n.

- Compute distances to the nearest medoid
  Line 9 of the algorithm involves computation of the distance from each point $x \in X - C$. It requires to compare all distances among each point x to all the medoids, so normally it requires $T_2(n, k) = (n - k) * k$. At this stage c_1 is the only medoid discovered, therefore $k = 1$.

$$T_2(n,k) = n - 1 \tag{5.50}$$

- Compute $c_2, \ldots, c_k$
  Final loop of the algorithm involves the computation of the medoids up to k. In this section we need to focus on the nested loop. We get:

$$\begin{aligned}
T_3(n,k) &= \sum_{l=2}^{k}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_1 + (n-l)l\right) \\
&= \sum_{l=2}^{k}\left(\alpha_1 n^2 + (n-l)l\right) \\
&= \sum_{l=2}^{k}\alpha_1 n^2 + \sum_{l=2}^{k} nl - \sum_{l=2}^{k} l^2 \\
&= \alpha_1 n^2(k-1) + n\left(\sum_{l=1}^{k} l - 1\right) - \left(\sum_{l=1}^{k} l^2 - 1\right) \\
&= \alpha_1 n^2(k-1) + n\left(\frac{k(k+1)}{2} - 1\right) - \left(\frac{k(k+1)(2k+1)}{6} - 1\right) \\
&\ldots \\
&= \alpha_1 n^2(k-1) + n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1
\end{aligned} \tag{5.51}$$

which we denote with α_1 the constant time required for the evaluation of the minimal distance from d_j and m_{ij}, and the assignment into c_l

Finally, we can evaluate total time spent from the algorithm summing up all the phases.

$$\begin{aligned} T(n,k) &= T_1(n,k) + T_2(n,k) + T_3(n,k) \\ &= \alpha_1 n^2 + n - 1 + \alpha_1 n^2(k-1) + n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1 \\ &= \alpha_1 n^2 k + n(\frac{k^2}{2} + \frac{k}{2}) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) \end{aligned} \tag{5.52}$$

Through the usage of the asymptotic notation we can classify the algorithm as:

$$O(n^2k + nk^2 - k^3) \tag{5.53}$$

5.3.3 Pseudo Code

In this section we present the code used in the implementation of the $PAMSILHOUETTE$ algorithm.

The algorithm 5.10 represents the pseudocode required for the execution of $PAMSILHOUETTE$. It requires only the dataset X as input, in order to produce as output the c_*, which represents a set of medoids. Associating each sample with the closest medoid, we obtain as results clustering with maximum value of Silhouette observed during its execution.
Initially it generates the distance matrix among all objects into X. It is required for $BUILD$. Then it defines some variables used later: S_* stores the best result of Silhouette obtained when the combination of medoid is c_* and λ_t is used as a stopping condition of the algorithm. It provides a threshold for the silhouette, in order to stop the execution when the algorithm produces clustering with a poor value of silhouette. It iterates over k, starting from a number of clusters equal to 2. S_k contains a temporary result of the best Silhouette obtained with k clusters. The algorithm is repeated for each k, until the stopping condition does not stop the execution. At each iteration it increases the number of k, it executed the $BUILD$ algorithm in order to detect the best initial medoid for a given k, and initialize to -1 current value of best Silhouette S_k.
For each k is executed another loop. Each iteration is determined from the variable t, which

Algorithm 5.10 PAMSILHOUETTE Algorithm

```
1: Input
2:     $X$    dataset
3: Output
4:     $c_*$    medoids, with produces clustering with miximum silhouette
5: procedure PAMSILHOUETTE($X$)
6:     $M \leftarrow \emptyset$
7:     for $i \leftarrow 1$ $to$ $n$
8:         for $j \leftarrow 1$ $to$ $n$
9:             $M_{i,j} \leftarrow d(x_i, x_j)$
10:    $\lambda_t \leftarrow \frac{17}{20}$
11:    $k \leftarrow 1$
12:    $S_*, S_k \leftarrow -1$
13:    $c_*$
14:    while $S_k \geq \lambda_t S_*$ $and$ $k < n - 1$
15:        $k \leftarrow k + 1$
16:        $t \leftarrow 0$
17:        $c_{t,k} \leftarrow BUILD(M, k)$
18:        $S_k \leftarrow -1$
19:        repeat
20:            $t \leftarrow t + 1$
21:            for $i \leftarrow 1$ $to$ $k$
22:                for all $x_j \in X - c_{t-1,k}$
23:                    swap $x_j$ and $c_i$
24:                    $labels$ $\leftarrow$associate each point to closest medoid
25:                    if $S_k \geq$ silhouette(x,labels)
26:                        $S_K \leftarrow$ silhouette(X, labels)
27:                        $C_t \leftarrow$ swapped medoids
28:        until $s_i > s_{i-1}$
29:        if $S_k > S_*$
30:            $S_* \leftarrow S_k$
31:            $c_* \leftarrow c_{t,k}$
32: return $c_*$
```

we increase at each iterations of this inner round. At each step it tests the current medoids: for each medoid c_i it swaps with a sample x_j, This operation is repeated for each sample into dataset X and for each medoid into c. At each swap it evaluates the Silhouette score gained through association of each sample to the closest medoid. If it detects a swap improving the current value of Silhouette S_k (when medoids were $c_{t-1,k}$), it generates a new set of medoids $c_{t,k}$ with swap applied. This iteration is executed until any swaps perform an improvement of the Silhouette. In this scenario the algorithm exits from repeat-until block, it updates if necessary the best value of Silhouette S_*, and relative medoids c_*, and continue with bigger k.
Given a specific k,at each iteration of the algorithm we would like to find medoids $c_{t,k}$ which increases the value of the silhouette computed when medoids were $c_{0,k}$. Assuming F as functions that assign Silhouette value to a given cluster, we obtain a monotonic sequence of value:

$$F(c_{0,k}) < F(c_{1,k}) < \dots \tag{5.54}$$

We need to elaborate a strategy to improve gradually the value of silhouette. In $PAMSILHOUETTE$, it is performed from the swapping of a not medoid point x_j with a medoid sample $c_{t,k}$. This produces a continued improvement of the goal function defined as Silhouette index.
Since it requires an $a - priori$ knowledge of the number of clusters required, we perform the algorithm for all the possible k. As an improvement of this trivial solution we set a threshold used as a stopping criteria of the algorithm. It is the same idea presented into the $extendeKMeans$.

5.3.4 Choice of Parameters

Analyzing the pseudo code available into Algorithm 5.10 we can notice that it is present a parameter λ_c useful for the stopping of the algorithm. As we perform into $extendeKMeans$ we introduce a parameter related to the best Silhouette value. The algorithm is executed until this condition is verified:

$$currentSilhouette > \lambda_c * bestSilhoutte \tag{5.55}$$

$currentSilhouette$ means the highest value we can obtain on Silhouette with a selected k, while $bestSilhouette$ is the best value of Silhouette among all the possible k. It is necessary to detect the best clustering in terms of Silhouette without requiring any input constraint. Value of λ_c, as explained into $extendeKMeans$, needs to be tuned to avoid a search for large value on k, with cluster with poor silhouette value. On the other hand, an early stopping can deny the possibility of an increasing Silhouette with large k. For this reason value is set to:

$$\lambda_c = \frac{17}{20} \tag{5.56}$$

5.3.5 Complexity and Correctness

In this last section we need to analyze in details time complexity of the algorithm proposed. The analysis involved the computation of the cost function for the main structure presented in the procedure.

- creation of matrix of distances M
  The first step of the algorithm is the creation of the matrix of distances M used later into $BUILD$ procedure.

  $$T_1(n,k) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_1 = \alpha_1 n^2 \tag{5.57}$$

  which evaluation of the distance between x_i and x_j is computed into constant time $O(1)$.

- BUILD
  Initial medoid are samples returned from $BUILD$ procedure, the cost function $T_2(n,k)$ related of this algorithm is highlighted in 5.52.

- Detection of best medoid for a given k (from line 19 to line 28)
  Core of the $PAMSILHOUETTE$ algorithm is the swap and the test of each point into X and into $c_{t,k}$. For each iteration it is necessary to compute the swap, in constant time $O(1)$, associate each point to the closest label (cost function related is $(n-k)k$) and compute Silhouette (cost function related is $O(n^2)$). It is fundamental to detect the possible number of repetitions into the loop.
  In order to repeat the loop we need to detect k medoids $\in$ $c_{t,k}$ which improves Silhouette value computed with medoids $\in$ $c_{t-1,k}$. In the worst case we get a slight improvement of each iteration. It is important to notice that it is not possible that we get at the end of an iteration same medoids of the beginning, because $S_t > S_{t-1}$. We create a monotonic sequence:

  $$S(c_0) < S(c_1) < \dots \tag{5.58}$$

Therefore in worst scenario we found at each step a new combination that increase Silhouette, and it is necessary to test them all. Possible combinations of k medoids, given dataset of n points, are:

$$\binom{n}{k} \tag{5.59}$$

Finally we compute the cost function for the whole loop.

$$\begin{aligned}
T_3(n,k) &= \binom{n}{k} \sum_{i=1}^{k} \sum_{j=1}^{n-k} \left(\alpha_1 + (n-k)k + n^2\right) \\
&= \binom{n}{k} \sum_{i=1}^{k} \left(\left(\alpha_1 + (n-k)k + n^2\right)(n-k)\right) \\
&= \binom{n}{k} \left(\left(\left(\alpha_1 + (n-k)k + n^2\right)(n-k)\right) k\right) \\
&\dots \\
&= \binom{n}{k} \left(n^3k + \alpha 1nk - 2nk^3 + k^4 - \alpha_1 k^2\right)
\end{aligned} \tag{5.60}$$

where we denote as α_1 the constant time required for the swap.

With the $T_1(n,k)$, $T_2(n,k)$ and $T_3(n,k)$ we obtain all tools required for calculation of cost function of $PAMSILHOUETTE$. We need to calculate the number of iterations of the while loop into line 14 in the worst case scenario. In this situation we have to iterate through all possible number of clusters, up to $n-1$.
We can express the total cost function as:

$$\begin{aligned}
T(n,k) &= T_1(n,k) + \alpha_1 + \sum_{k=2}^{n-1} T_2(n,k) + \alpha_2 T_3(n,k) \\
&= \alpha_1 n^2 + \alpha_1 + \sum_{k=2}^{n-1} (\alpha_1 n^2 k + n(\frac{k^2}{2} + \frac{k}{2}) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + \\
&\qquad + \alpha_2 \binom{n}{k} \left(n^3k + \alpha 1nk - 2nk^3 + k^4 - \alpha_1 k^2\right))
\end{aligned} \tag{5.61}$$

In 5.61 we noted as α the constant terms. For an asymptotic notation we define some bound for an easier computation of the cost function. Recalling that $k < n$ is always valid and:

$$\binom{n}{k} \approx O(n^k) \tag{5.62}$$

we get:

$$\begin{aligned} T(n,k) \leq \alpha_1 n^2 + \alpha_1 + \sum_{k=2}^{n-1} (\alpha_1 n^2 k + n(\frac{k^2}{2} + \frac{k}{2}) + \\ + \alpha_2 n^k (n^3 k + \alpha 1 n k + k^4)) \end{aligned} \tag{5.63}$$

Concluding we define the asymptotic complexity fo the $PAMSILHOUETTE$ algorithm.

$$O(n^n) \tag{5.64}$$

The last analysis involves a discussion on the termination of the algorithm. Since it is not possible to detect if $PAMSILHOUETTE$ performs better or worse than PAM algorithm, we perform evaluation on the termination of the algorithm. Even in this case it is an iterative algorithm, so it is trivial to detect that all the variables constituting the loop do not mutate during each iteration.
It is important to detect if the repeat until block reaches its conclusion.
For a specific number of cluster k, suppose that it exists at least one set of medoids $c_{*,k}$ which generates the best possible value of Silhouette S_*. We recall that since we detect k medoids, there is a unique association through all the sample and medoids. We start from medoid $c_{0,k}$ which generates the value of silhouette S_0. At each iteration we obtain a clustering $c_{t,k}$ with the increasing value of silhouette S_1. We create the monotonic sequence:

$$S(c_{0,k}) < S(c_{1,k}) < \cdots < S(c_{*,k}) \tag{5.65}$$

The sequence is upper bounded from the best possible clustering. Although it is delimited we cannot determine if it finishes before reaching it, because the algorithm may lead to stop with $c_{i,k}$ with no more swap that can improve Silhouette, even if $S(c_{i,k}) < S(c_{*,k})$.
Since each iteration is associated with a clustering value, we cannot be stuck into a repetition of the same clustering more than once. We conclude that it exists a clustering formed from $c_{i,k}$ which $S(c_{i,k}) \leq S(c_*)$ and no possible improvement of Silhouette, forcing the algorithm to finishes.

5.4 ExtendePAM

The final algorithm proposed into this thesis is a mixture of the idea presented in the previous algorithms. Except from the first naive approach presented on this thesis, $bruteForce$, some features of the $extendeKMeans$ and $PAMSILHOUTTE$ approaches can be mixed together in order to derive this new algorithm, $extendePAM$.

5.4.1 Description of the Approach

This algorithm executes mainly the procedure involved and presented into $extendeKMeans$, but into $extendeKMeans$ we adopt $k - Means$ algorithm initially to define the first labels of the samples, as depicted into Algorithm 5.4 on line 11. Here, we use $PartioningAroundMedoid$ algorithm for initial clustering.
Once we execute this algorithm we adopt the same technique and strategies used into $extendeKMeans$. This approach is presented as a completion of the thesis, in order to inspect all the possible strategies that we can achieve trying to improve existing algorithm.

5.4.2 Pseudo Code

In the pseudo code presented into this section we report only the different procedure than $extendeKMeans$. Even if the algorithm requires external procedures, they are identical to the one provided into $extendeKMeans$, which pseudo code is available into Algorithm 5.5, 5.6 and 5.7 As $extendeKMeans$, it is formed from a main function described into Algorithm 5.11.

Algorithm 5.11 Extension of Partitioning Around Medoid

```
1:  Input
2:      $X$    dataset
3:  Output
4:      $C_*$    clustering with better silhouette index
5:  procedure EXTENDEPAM($X$)
6:      $C_* \leftarrow \emptyset$
7:      $S_*, S_{-1} \leftarrow 0$
8:      $k \leftarrow 2$
9:      $\lambda_t \leftarrow \frac{17}{20}$
10:     while $S_{-1} \geq \lambda_t S_* \ and \ k < n - 1$
11:         $C_k \leftarrow PAM(X, k)$
12:         $\psi_0, \psi_1 \leftarrow min(silhouette(C_k)), max(silhouette(C_k))$
13:         $\lambda_c \leftarrow \frac{10(\psi_1 - \psi_0)}{log K + 130} + \psi_0$
14:         $M_k \leftarrow generateMPoint(X, C_k, \lambda_c)$
15:         $T_k \leftarrow closestCenter(M_k, C_k, k)$
16:         $S_k, C_k \leftarrow bruteForce(X, T_k, C_k)$
17:         if $S_k > S_*$
18:             $S_* \leftarrow S_k$
19:             $C_* \leftarrow C_k$
20:         $k \leftarrow k + 1$
        return $C_*$
```

The main difference from Algorithm 5.4 is the usage of Partitioning Around Medoid algorithm (PAM). For an easier comprehension, we proposed the meaning of the notation presented. With C_* we denoted the best clustering, which corresponds with the clustering that maximizes value of Silhouette. S_* and S_{-1} contain the best Silhouette achieved and the value computed on C_* and the initial Silhouette, respectively. λ_c and λ_t highlights the threshold required for the algorithm, used for point selection and stopping criteria for the algorithm.
Core of the algorithm is the loop from line 10 up to line 20. It is executed until the best silhouette achieved for the specific k, S_k is bigger than stopping criteria; the condition involves also the number k of the clusters, since $0 < k < n$.
At each iteration k is used to store the current number of clusters. C_k saves the clusters

returned from $PartitioningAroundMedoid$ Algorithm. M_k is a set containing all the samples which they will be reclassified later. For this purpose, it is necessary to define the threshold λ_c. Through the usage of M_k we define T_k with $closestCenter$ procedure. Here for each sample into M we associate labels to the closest two clusters. For a given T and sample $x_i \in M$, we get $T_i[0]$ the value of the closest cluster, while $T_i[1]$ contains the second closest cluster. Finally S_k, C_k are the Silhoutte and cluster obtained from $bruteforce$ procedure, it tests all the possible combination into T_k to produce the best clustering, one that maximizes silhouette. Finally, it updates if required the best Silhouette into S_*.

5.4.3 Complexity and Correctness

Now we focus on the detection of the asymptotic complexity in time of the algorithm. Since it executes procedure available from $extendeKMeans$ algorithm, for the computation of cost function we use some results obtained into section 5.2.4.
It is required an analysis of asymptotic complexity of $PartitioningAroundMedoid$ algorithm, described into Algorithm 2.4. In this pseudo code the initial medoids are chosen uniformely at random from the dataset, but into $PAMSILHOUETTE$ we adopt a specific seeding technique, $BUILD$. We perform an analysis of cost function of the PAM obtained after replacing into line 7 of Algorithm 2.4, the $BUILD$ algorithm, as exploited in line 17 of Algorithm 5.10. We divide evaluation of the cost function of the algorithm into different portions of the pseudo code. Later, we combine these results in order to detect the final asymptotic complexity. Thenput of the cost function are dataset X and number of clusters k.

- BUILD initial seeding
  The evaluation of the cost function returned from the $BUILD$ algorithm is exploited into 5.52. Here we propose the same approach returning the same cost function.
- Associate each point to the closest medoid
  Even if the pseudo code is not specific related to the technique used to compute the association, we can evaluate the cost function as:

$$T_2(n,k) = \sum_{i=1}^{n-k}\sum_{j=1}^{k}\alpha_1 = \alpha_1(n-k)k \tag{5.66}$$

  where we denoted with α_1 the constant time required for evaluate of distance function. The formula derives from assumption that each point $\notin M$ need to be compared with all samples $\in M$ in order to detect closest medoid.

- Cost function for each iteration
  First, it is useful to detect the cost function associated for each iteration of the while loop into the pseudo code. We need to define the cost associated to a specific medoids configuration. Cost depicted into the algorithm refers to the sum of distance from each samples to the closest pseudocode. For this reason, cost can be computed during the association to each sample with closest medoid, without affecting $T_2(n,k)$.
  For each iteration we get:

$$T_3(n,k) = \sum_{i=1}^{k}\sum_{j=1}^{n-k} \alpha_1 + \alpha_2(n-k)k = \alpha_2(n-k)^2k^2 + \alpha_1(n-k)k \quad (5.67)$$

  Into 5.67 we indicates with α_1 operation executed into contant time. Moreover we use 5.66 when we refer to the cost of associating each sample to closest medoid.

To build the whole cost function of Algorithm 5.11, we need to estimate the number of iterations in worst case scenario. Once we notice that cost of configuration has to increase at each loop, we easily conclude that set of medoid is different at each iteration, and a set cannot be used twice. The worst case involved a slightly improvement of medoid at each iteration. Assuming as M_*, the set of medoids which minimizes cost function, in the worst case we generate a sequence of medoids.

$$M_0 < M_1 < \cdots < M_* \quad (5.68)$$

Since it exists an upper bound from the sequence we derive that the algorithm always finishes, and in the worst case we get a new set of medoids at each iteration. Worst case is testing of all possible $k - subset$ from X samples, hence:

$$\binom{n}{k} \quad (5.69)$$

We can define the cost function of $PartitioningAroundMedoid$.

$$\begin{aligned} T_4(n,k) &= T_1(n,k) + T_2(n,k) + \binom{n}{k}T_3(n,k) \\ &= \alpha_1 n^2(k-1) + n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1 + \\ &\quad + \alpha_1 nk - \alpha_1 k^2 + \binom{n}{k}(\alpha_2(n-k)^2k^2 + \alpha_1(n-k)k) \end{aligned} \quad (5.70)$$

Recalling from 5.52 the cost function $T_1(n,k)$ of BUILD algorithm and:

$$\binom{n}{k} = O(n^k) \tag{5.71}$$

we get asymptomatic complexity of PAM Algorithm.

$$O(n^{k+2}) \tag{5.72}$$

Once we compute the analysis of PAM we can proceed with the evaluation of the $extendePAM$ algorithm. Since it uses some procedures available into $extendeKMeans$, we can recall the cost function of $generateMPoint$, $closestCenter$ and $bruteForce$, where cost function is available into 5.30, 5.31 and 5.32 respectively. With the cost function discovered into 5.70, we can calculate the function related to the cost of each iteration of $extendePAM$. We need to introduce the cost function related to the calculation of the threshold λ_c, which we denoted with $T_5(n,k)$. Even in this case we need to assume the number of iterations with reference to worst case scenario. Since the change of initial algorithm from $K - Means$ to $PartitioningAroundMedoid$ does not affect the number of iterations involved, we can assume the same scenario of $extendeKMeans$, hence it needs to test all the possible k, from 2 up to $n-1$. Finally we get:

$$T(n,k) = \sum_{k=1}^{n-1} \alpha_1 + T_1(n,k) + T_2(n,k) + T_3(n,k) + T_4(n,k) + T_5(n,k) \tag{5.73}$$

where we indicate with α_1 the constant time required for assignment. Substituting the different cost function we obtain:

$$\begin{aligned} T(n,k) = & \sum_{j=1}^{n-1} \alpha_1 + \alpha_1 n^2 + \alpha_2 n + \alpha_1 n + \alpha_2 k\,|M| + \\ & + 2^{|M|}(|M|\,\alpha_1 + \alpha_2 n^2 + \alpha_3) + \\ & + \alpha_1 n^2(k-1) + n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1 + \\ & + \alpha_1 nk - \alpha_1 k^2 + \binom{n}{k}(\alpha_2(n-k)^2k^2 + \alpha_1(n-k)k) + \\ & + \alpha_1 n^2 + \alpha_2 n \end{aligned} \tag{5.74}$$

It is trivial to detect that the asymptotic complexity in 5.74 is related to $2^{|M|}$. Into the worst case scenario set M can potentially include all the possible points, defining the asymptotic complexity of $extendePAM$ in:

$$O(2^n n^{k+3}) \tag{5.75}$$

Finally we discuss about the correctness of the algorithm.

Proposition 3. *Given a dataset of n points $X = \{x_1, \ldots, x_n\}$, the application of Algorithm 5.11 returns a value of Silhouette S_* generated from a clustering C_*. We get $S_k \leq S_* \quad \forall k$, where S_k is the clustering returned from PAM with k initial medoids. Furthermore, Algorithm 5.11 always finishes.*

Proof. Proof of the Proposition 3 can be derived from Proposition 2, since results of the proof does not depends on the labels associated to the initial algorithm applied. □

5.5 Improved version of extendePAM, extendePAMv2

The last approach proposed aims to improve the previous algorithm discovered, $extendePAM$. As $extendeKMeansv2$ performs an improvement for $extendeKMeans$, even in this case we are looking for an improvement in time from the first version of the algorithm. The strategies implemented recall $extendeKMeansv2$, in fact we change the construction of the set M. In the basic version of the algorithm, the selection of point into M is done by a threshold. If a sample has a value below threshold it will be stored into M and reclassified later into $bruteForce$ procedure. Here we save the worst $log_2(M)$ points. Worst is referred to the Silhouette Value associated to each sample. It can be described with the application on line 14 of Algorithm 5.11 of the procedure $generatePointM2$, described into Algorithm 5.8. We obtain the best performance in term of asymptotic complexity.

5.5.1 Complexity and Correctness

Since we start from the Algorithm 5.11, we can adopt its cost function, recalling to change $T_1(n,k)$ into 5.73. In fact, $T_1(n,k)$ contains cost function associated with $generateMPoint$, while into this algorithm we need to replace that formula with the equation in 5.38. It consists of an analysis made on cost on procedure $generateMPointv2$. Finally we get:

$$\begin{gathered}
T(n,k) = \sum_{j=1}^{n-1} \alpha_1 + \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n + \alpha_1 n + \alpha_2 k\,|M| + \\
+ 2^{|M|}(|M|\,\alpha_1 + \alpha_2 n^2 + \alpha_3) + \\
+ \alpha_1 n^2 (k-1) + n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1 + \\
+ \alpha_1 nk - \alpha_1 k^2 + \binom{n}{k}(\alpha_2 (n-k)^2 k^2 + \alpha_1 (n-k)k) + \\
+ \alpha_1 n^2 + \alpha_2 n
\end{gathered} \tag{5.76}$$

In 5.76 we know exactly the cardinality of the set M, hence we can substitute its value into the equation, obtaining:

$$\begin{gathered}
T(n,k) = \sum_{j=1}^{n-1} \alpha_1 + \alpha_1 n^2 + \alpha_2 nlogn + \alpha_3 log_2 n + \alpha_1 n + \alpha_2 k\,|M| + \\
\alpha_1 n log_2 n + \alpha_2 n^3 + \alpha_3 n + \alpha_1 n^2 (k-1) + \\
+ n(\frac{k^2}{2} + \frac{k}{2} - 1) - k(\frac{k^2}{3} + \frac{k}{2} + \frac{1}{6}) + 1 + \\
+ \alpha_1 nk - \alpha_1 k^2 + \binom{n}{k}(\alpha_2 (n-k)^2 k^2 + \alpha_1 (n-k)k) + \\
+ + \alpha_1 n^2 + \alpha_2 n
\end{gathered} \tag{5.77}$$

We can conclude finally with the asymptotic complexity of the $extendePAMv2$ algorithm:

$$O(n^{k+3}) \tag{5.78}$$

6
Experimental evaluation

Inside this chapter we evaluate and discuss the results achieved implementing the algorithm described into the previous section. Each result will be compared with other techniques and with already existing clustering algorithms. Besides the Silhouette value we print also time required for the completion of the algorithm. Since it is related with the power of machine, we perform the experiments on the same computer in order to not affect the measures.

6.1 Dataset

For the execution of the algorithm, in order to produce a relevant result to be compared, we need to identify a common environment that it will be reproduced in each test that we execute.

The dataset used in this thesis will be the Iris Dataset available into Sckit Learn library of Python, Breast Cancer Wisconsin Data Set and banknote authentication Data Set, both available from UCI Machine Learning Repository [26][27].

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. A scatter plot of the sample recorded into the dataset is available into Figure 6.1.

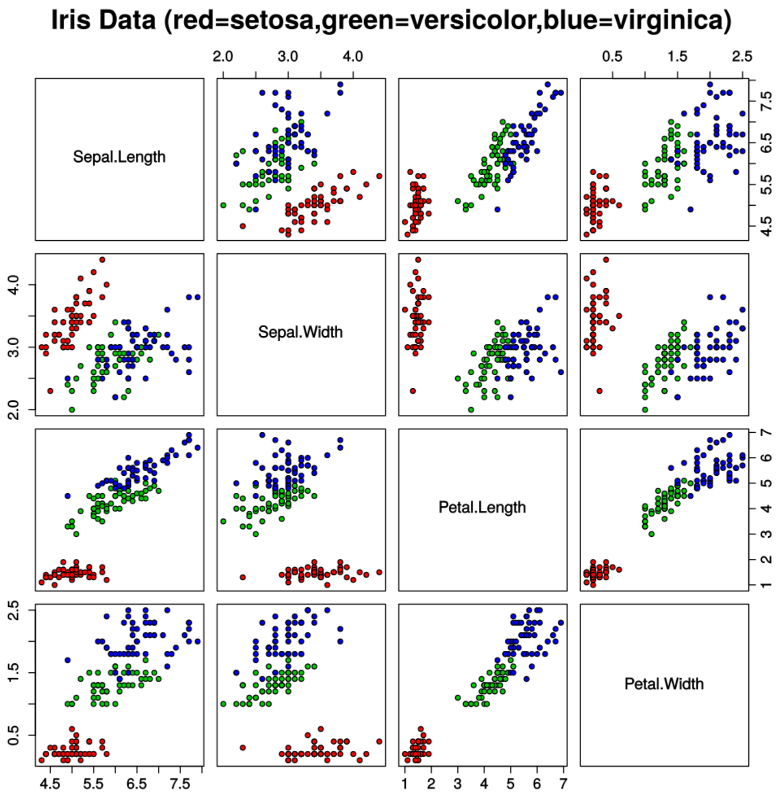

Figure 6.1: Iris Dataset. Each plot shows distribution of the sample filtering for only 2 features. Taken from [5].

Purpose of the algorithm will be a proper classification of the sample in clustering. It is worth pointing out that the goal is the detection of the clustering that maximise the Silhouette, regardless if its related K will be different from the real value of 3.

Into the Brute Force approach we will force to execute the algorithm on a small subset of the dataset due to time required to the completion of the algorithm on big dataset. We select randomly 13 index and they will be used as input of the algorithm. These values will be used in each implementation in order to provide a comparison between the algorithm proposed and the brute force approach. A plot of distribution of classes into these samples is available into Figure 6.2

Breast Cancer Wisconsin (Original) Data Set contains a dataset with samples that arrive periodically as Dr. William H. Wolberg, which reports his clinical cases. This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison. It is composed of 10 different features for each sample, which it is classified into two classes, value 2 for be-

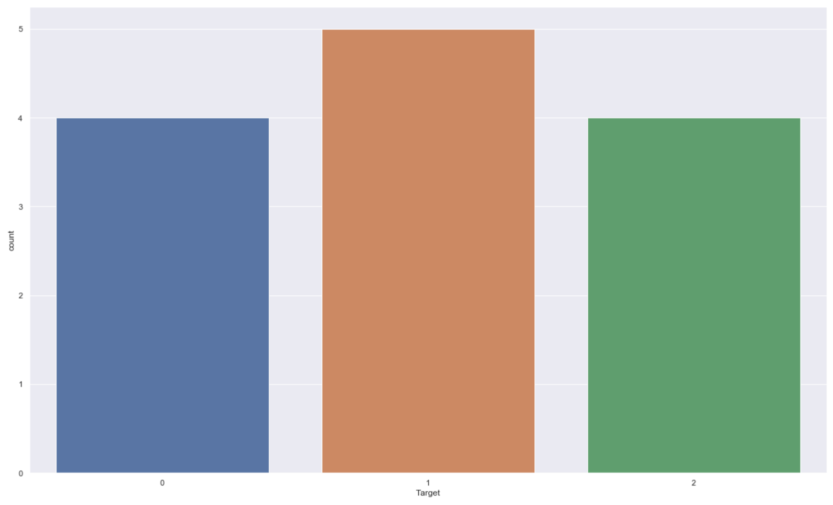

Figure 6.2: Reduced Iris dataset. 1,2 and 3 is used to labelling three classes of iris

nign, 4 for malignant. The attribute informations are the following:

- 1 - Sample Code Number - Domain: id number
- 2 - Clump Thickness - Domain: 1 - 10
- 3 - Uniformity of Cell Size - Domain: 1 - 10
- 4 - Uniformity of Cell Shape - Domain: 1 - 10
- 5 - Marginal Adhesion - Domain: 1 - 10
- 6 - Single Epithelial Cell Size - Domain: 1 - 10
- 7 - Bare Nuclei - Domain: 1 - 10
- 8 - Bland Chromatin - Domain: 1 - 10
- 9 - Normal Nucleoli - Domain: 1 - 10
- 10 - Mitoses - Domain: 1 - 10

Samples arrive periodically at professor Wolberg, which reports, from January 1989 to November 1991, 699 samples into dataset. In this thesis this dataset is filtered from first feature related to the code number and from samples with some missing value, labelled in the dataset with '?' character. Furthermore, it is held only the unique instances, in order to delete duplicates. Results are a dataset of 449 samples, each one described with 9 features.

Into the Brute Force approach we reduce the number of samples from dataset in order to obtain results in reasonable time. As we perform into Iris Dataset, from 449 samples, we select 13 data points, which it is shown into Figure 4.3 the counting of different classes for reduced dataset.

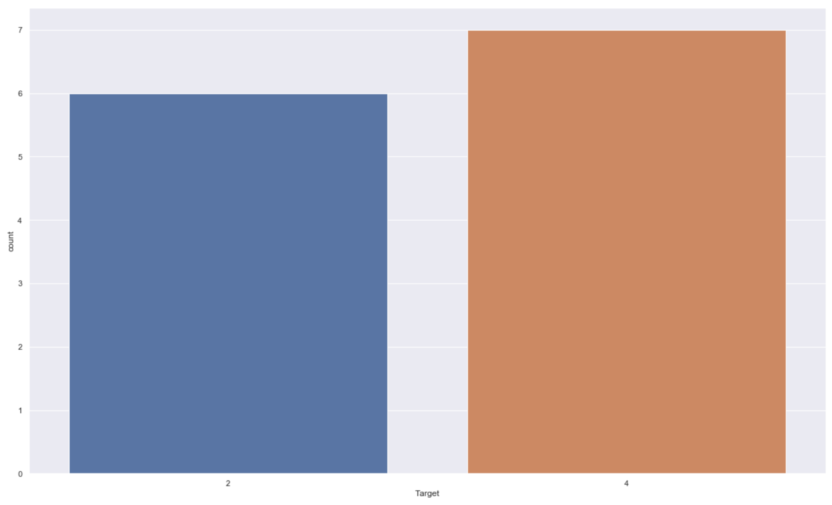

Figure 6.3: Reduced Breast Cancer Wisconsin Data Set. 2 and 4 is used to labelling two kinds of cancer, benign or malignant

Final dataset used into this thesis is the banknote authentication Data Set. It is the largest of the 3 datasets proposed into this thesis. It is designed for distinguishing genuine and forged banknotes. Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. A Wavelet Transform tool was used to extract features from these images. It is composed from 1372 sample, with 4 features:

- variance of Wavelet Transformed image - Continuos Value
- skewness of Wavelet Transformed image - Continuos Value
- curtosis of Wavelet Transformed image - Continuos Value
- entropy of image - Continuos Value

For the results dataset is filtered from duplicated values, resulting 1348 samples. The real label associated to the dataset is subdivided into two classes, in order to distinguish if a banknote is authentic or forged. As in the previous dataset we generate also in this cases a reduced version of dataset, composed by 13 samples, which is used first into Brute Force approach, and later it is also executed from all other techniques. Data extracted from dataset has real label distributed as highlighted into Figure 6.4.

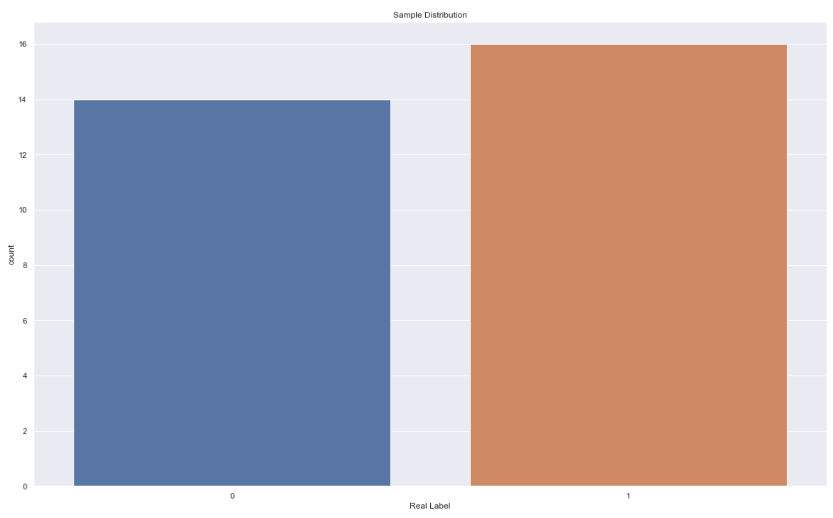

Figure 6.4: Reduced banknote authentication dataset. 0 and 1 is used to labelling two different classes of banknote

6.2 Implementation

Into this chapter we provide some experimental results obtained after implementation of the algorithm provided only as pseudocode in the chapter before. Into this thesis we decide to implement them by the usage of Python as programming language. Most of tools required are provided natively with the most famous library.
For clustering purpose, Python provides some powerful libraries which they include all the tools required for algorithm implementation. Main libraries are the following:

- Scikit-learn
  It is a free software machine learning library for the Python programming language [28, 26]. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. For the specific purpose of the thesis it is used to perform

algorithms like $HierachicalClustering$ and $K-Means$. Furthermore, it provides a native procedure for calculation of Silhouette index for each sample.

- pyclustering
  It is a Python data mining library (clustering algorithm, oscillatory networks, neural networks)[29].It supports a wider range of clustering technique, including PAM, which it is widely used into this thesis.

- more-itertools
  In more-itertools there are additional building blocks, recipes, and routines for working with Python iterables. In this case we have to deal with a useful library for multiple purposes. It provides in fact some basic functionalities. In this thesis we can find into this library procedure $algorithm_u$, which we used into $bruteForce$ algorithm to perform all the $k - subset$ of a given dataset.

6.3 Comparison of commonly used Clustering Algorithms

The comparison involves the main clustering algorithms described in the chapter 2. The proposed algorithms are:

- $k - Means$
- $k - Means$ with the application of $k - Means++$ for initial seeding
- $k - Medoid$
- Hierarchical Clustering

For the Hierarchical clustering we provide results for all the linkage criteria defined in section 2.6. Since each algorithm can produce different results at each iteration, tests are executed for $t = 100$ iterations. Final results have shown the average value returned from the test and the maximum silhouette achieved. Since the objective of the thesis is the detection of the best silhouette it is interesting the best silhouette that an algorithm can achieve, even if some circumstances are required.
Another important objective of the thesis is the detection of the number of cluster k which clustering provides the best silhouette. Since all the techniques tested here require k as the input of the problem(or it can be a stopping condition into hierarchical clustering), we perform test for k from 2 up to $|X| - 1$. Table 6.1 shows the result for an interesting value of K,

in this scenario the value starts decreasing after $k = 7$, so it is useless for the thesis a further k. All the tests are executed on Iris Dataset. We perform the comparison with the iris dataset[4].

# of Clusters	2	3	4	5	6	7
K-Means++	0.6810	0.5276	0.4304	0.3905	0.3716	0.3551
K-Medoid	0.6858	0.5400	0.4698	0.4150	0.3603	0.3360
HC - Single	0.6867	0.5121	0.2819	0.2838	0.2214	0.1328
HC - Complete	0.5160	0.5136	0.4998	0.3462	0.3382	0.3298
HC - Average	0.6867	0.5542	0.4720	0.4307	0.3420	0.3707
HC - Ward	0.6867	0.5543	0.4890	0.4844	0.3592	0.3422

Table 6.1: Comparison of Average Silhouette value on Iris Dataset from most famous clustering algorithm

The table 6.1 shows the average value returned by the algorithm. The Hierarchical Clustering (HC into table 6.1), once the linkage criteria is defined, chose cluster to be merged into a deterministic way. The result is the same for all the iteration. For a graphic visualization, Figure 6.5 provides a simple comparison.

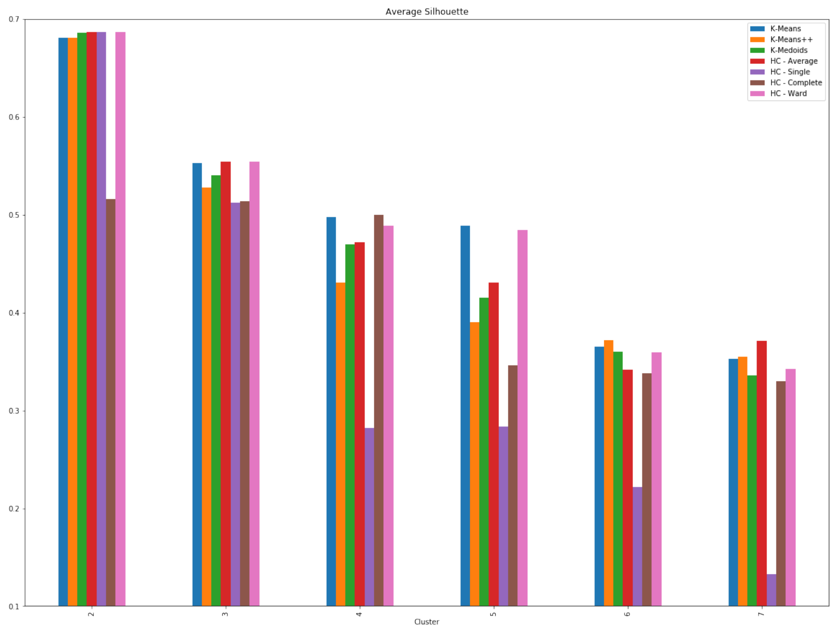

Figure 6.5: Histogram of Average Silhouette value on Iris Dataset from most famous clustering algorithm

For a better comparison of the algorithm, since the objective of the thesis is the maximization of silhouette, regardless if more execution occurs, here we also provide the best Silhouette value. We notice, for the aforementioned cause, that the maximum silhouette of the Hierarchical Clustering does not differ from the average one.

# of Clusters	2	3	4	5	6	7
K-Means	0.6810	0.5528	0.4981	0.4931	0.3687	0.3582
K-Means++	0.6810	0.5512	0.4973	0.4629	0.3828	0.3675
K-Medoid	0.6858	0.5553	0.4951	0.4933	0.4276	0.4429
HC - Single	0.6867	0.5121	0.2819	0.2838	0.2214	01328
HC - Complete	0.5160	0.5136	0.4998	0.3462	0.3382	0.3298
HC - Average	0.6867	0.5542	0.4720	0.4307	0.3420	0.3707
HC - Ward	0.6867	0.5543	0.4890	0.4844	0.3592	0.3422

Table 6.2: Comparison of maximum Silhouette value on Iris Dataset from most famous clustering algorithm

Into previous tables we highlight the box containing the best value achieved for each algorithm on the Silhouette value. Figure 6.6 reports histogram for a comparison of the dif-

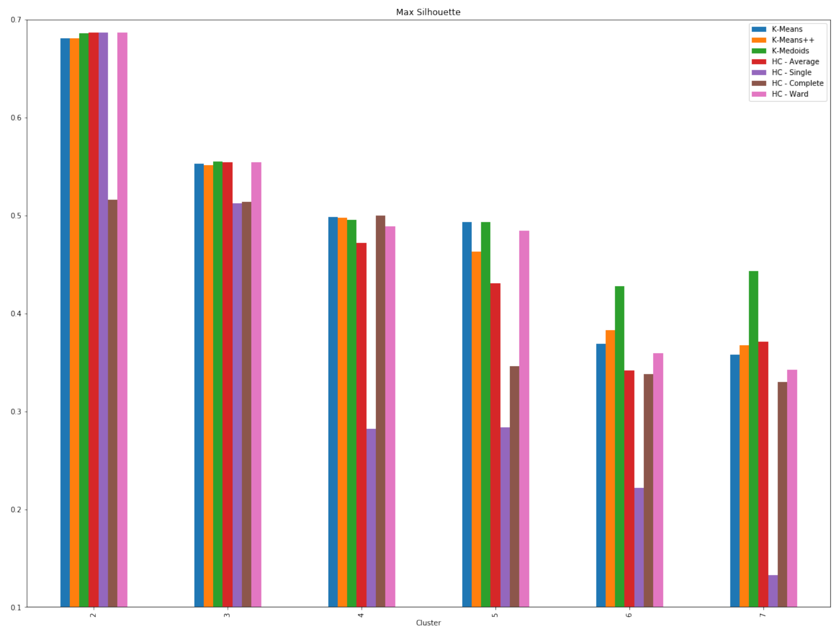

Figure 6.6: Histogram of maximum Silhouette value on Iris Dataset from most famous clustering algorithm

ferent silhouette among algorithms. When inspecting the value of the maximum silhouette

obtained from the algorithm, we notice that $K - Medoid$ algorithm is able to perform the best result also when k starts increasing. For small value of k, Silhouette are almost similar for each algorithm analyzed. On the other hand, $k - Medoid$ really suffers the initial seeding of the algorithm. On this comparison the initial medoids are chosen uniformly at random from the dataset. It is relevant to notice that a proper seeding initialization of $k - Medoids$ can generate clustering with higher value of Silhouette than $k - Means$. This is the motivation that prompted us to extend this technique into one algorithm. Moreover, there is ahigh variance of the results because the initial seeding is solved with appliance of $BUILD$ algorithm.
This analysis is prompt as a report on the best algorithm for Silhouette. The decision made into chapter 5 are related to the value gained into this comparison. It is also powerful for the analysis of the final silhouette value achieved from the algorithm presented in this thesis, to highlight their effectiveness.

6.4 Brute Force Approach

The first algorithm implemented was $bruteForce$. As highlighted from its asymptotic complexity it suffers on big dataset and for this reason the algorithm is performed only on few selected indexes from both datasets. All the other techniques will be also tested with these reduced versions of dataset, in order to highlights if they can achieve best result.
The brute Force algorithm produces as output only the best clustering with its silhouette value. By exploiting the $k - subset$ we can perform the best clustering for each different value of k. We print all the different Silhouette values computed on the reduced version of the dataset presented in sections 6.3.

We highlight the results of the best Silhouette returned by the algorithm for each dataset. As we consider the same number of samples from each dataset, the analysis in time loses the effectiveness repeated for all the dataset. We report into Table 6.4 the results obtained for the Reduced Iris Dataset, in order to provide an idea of the time elapsed for a dataset of only 13 samples.

Finally we display into Figure 6.7,6.8 and 6.9 an histograms where we highlight the different silhouette value achieved according to the number of clusters. Into each figure it is also reported the time required for the completion of algorithm for the given k. As described above we notice that the curve of time is always the same. We can notice that k ranges from the minimum value of 2 until the maximum value of $|X| - 1$. It is important to highlight

# of Clusters	Iris*	Breast Cancer*	Banknote*
2	0.6791	0.7396	0.7388
3	0.5562	0.7844	0.6038
4	0.5041	0.7163	0.6321
5	0.4742	0.7170	0.5897
6	0.4731	0.6852	0.5723
7	0.4072	0.6231	0.5277
8	0.3608	0.5656	0.4767
9	0.2922	0.4701	0.4130
10	0.2115	0.3733	0.3450
11	0.1149	0.2605	0.2080
12	0.0685	0.1331	0.1366

Table 6.3: Silhouette Results performed from BruteForce algorithm on reduced version of Dataset

the time required just for the execution of the algorithm with 13 samples. We got in total, an execution time of more than five hours. It provides once again a proof of unserviceability into the real life.

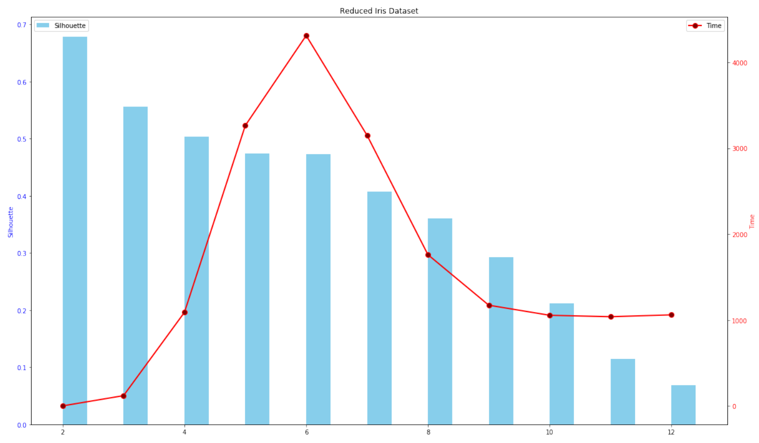

Figure 6.7: Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Iris Dataset

# of Clusters	Time	# of k-subset
2	1.8622	4092
3	120.9080	261625
4	190.0248	2532530
5	3263.8143	7508501
6	4310.8248	9321312
7	3149.9193	5715424
8	1760.8264	1899612
9	1173.1307	359502
10	1055.9288	39325
11	1039.3648	2431
12	1061.0524	78

Table 6.4: Evaluation of time required and number of k-subsets for each number of cluster performed on Reduced version of Iris Dataset

6.5 Extende K-Means and Improved version

6.5.1 Reduced Dataset

We analyze the performance provided by the first heuristic algorithm proposed, $extendeKMeans$ and its improved version. First we plot the results achieved by executing the algorithm on a reduced version of dataset. The number of K shown in the following results are due to the threshold set that provides a stopping condition on the algorithm. Here we analyze a value returned on Silhouette after the appliance of the reduced version of the dataset. These values can be compared with results obtained into the previous section, in order to evaluate effectiveness of this approach.

# of initial clusters	k-Means	extende KMeans	# of final clusters	extende KMeansv2	# of final clusters
2	0.6607	0.6607	2	0.6607	2
3	0.5520	0.5520	3	0.5520	3
4	0.5041	0.5041	4	0.5040	4

Table 6.5: Evaluation of silhouette value obtained after execution of $extendeKmeans$ and its improved version on reduced Iris Dataset

The table highlights the results of the algorithm shown at the end of its execution, in

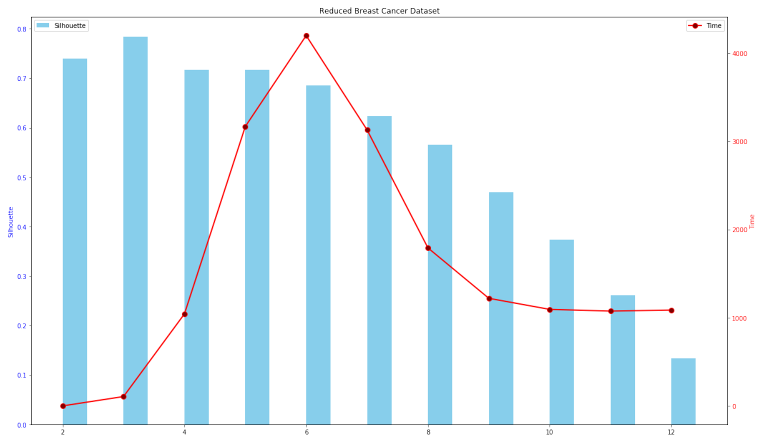

Figure 6.8: Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Breast Cancer Dataset

fact normally all the intermediate steps are not provided as output. We insert into the table 6.5, 6.6 and 6.7 column related to the number of clusters obtained into the best result of the $extendeKMeans$ and $extendeKMeansv2$. Due to the intrinsic composition of the algorithm, some cases may lead to a reduced number of clusters into output. This circumstance is invoked when all the points of a clusters are stored into M. During the testing of the possible combination into $bruteForce$, they can be all assigned to another cluster, reducing the number of clusters into the final results.

In this case we notice, that even if dataset is quite small we obtain a lower value compared to the bruteForce approach for the number of clusters 3 and 4. We are still able to detect the clustering which provides the best Silhouette in absolute term. It is due to $k-Means$ algorithm, in fact during the execution, $extendeKMeans$ and its improved version are not able to detect an improvement from the initial score of Silhouette.

While in this case we obtain the same number of clusters into the final clustering, we obtain a reduced number analyzing Table 6.6. It shows the results after the appliance of the algorithms on Breast Cancer Dataset.

Into this dataset we notice a reduction of the number of clusters into both algorithms when k is set to 4. This is due to M that contains an entire cluster. While the Silhouette achieved on reduced Iris is not the best, here we get the maximum result of Silhouette.
For the analysis of these reduced datasets we introduce a slightly different stopping condition,

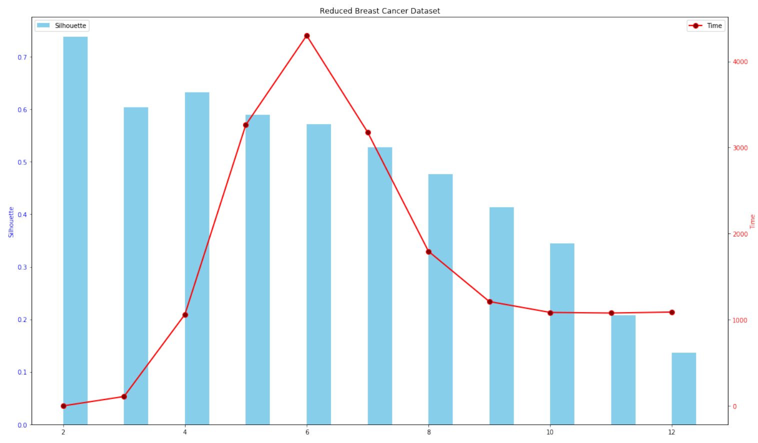

Figure 6.9: Histogram of Silhouette value and time required in seconds for execution of $bruteForce$ on Reduced Banknote Dataset

# of initial clusters	k-Means	extende KMeans	# of final clusters	extende KMeansv2	# of final clusters
2	0.7396	0.7396	2	0.7396	2
3	0.7844	0.7844	3	0.7844	3
4	0.7163	0.7844	3	0.7844	3

Table 6.6: Evaluation of silhouette value obtained after execution of $extendeKmeans$ and its improved version on reduced Breast Cancer Dataset

replacing 5.4. In fact we consider:

$$\begin{aligned} &SilhouetteValue(currentK) < \lambda_t maxValueSilhouette\ and \\ &currentNumberOfCluster = initialNumberOfClusters \end{aligned} \tag{6.1}$$

This variation is not relevant when we analyze the entire versions of the dataset because it is a very rare case the deletion of a cluster, instead a reduction of k is often performed into a small dataset.
For this reason the algorithm executed into Table 6.6 stops after the fourth iteration.

We report into Table 6.7 the results on the Silhouette obtained into the reduced version of the Banknote dataset. Even in this case we obtain the best possible Silhouette value, but we lose effectiveness in terms of Silhouette when the number of clusters start increasing. With $k = 3$ the result is lower than $bruteForce$ algorithm. All of the results do not show an improvement of the original value of $k - Means$, they are conditioned for the execution on

# of initial clusters	k-Means	extende KMeans	# of final clusters	extende KMeansv2	# of final clusters
2	0.7388	0.7388	2	0.7388	2
3	0.5180	0.5180	3	0.5180	3

Table 6.7: Evaluation of silhouette value obtained after execution of extendeKMeans and its improved version on reduced Banknote Dataset

the first algorithm, that it generates C_k. Into this analysis we do not include results on time, due to the size of dataset.

6.5.2 Complete Dataset

We focus here on the experimental results obtained from the execution of $extendeKmeans$ and $extendeKMeansv2$ on the entire version of the Dataset. Table 6.8 refers to Iris Dataset.

# of Clusters	2	3	4
K-Means	0.6810	0.5528	0.4981
Time K-Means	0.0120	0.0200	0.0230
extendeKMeans	0.6858	0.5543	0.4981
Time extende.	0.0260	0.0399	0.0330
extendeKMeansv2	0.6867	0.5553	0.4981
Time extendeV2	0.2757	0.3037	0.3067

Table 6.8: Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Iris Dataset

Analysis made on the entire version of the dataset showing that best result of Silhouette is obtained with $k = 2$. Into the Iris dataset, we retrieve the optimum result after the appliance of $extendeKMeansv2$ algorithm, while the original $k - Means$ and the first version $extendeKMeans$ are not able to detect such a good result. Also, for the other value of k we obtain the best results during the execution of $extendeKMeansv2$. It is due to the threshold used to the selection of M. It generates the set M with $|M| < log_2(n)$.
The algorithms provided in the chapter 5 provide as results only the best value achieved for each algorithm. All tables available into this section are obtained by exploiting the results of the algorithm after each iteration.
Into the reduced version of the Dataset we notice that it is possible to obtain a reduced number of clusters as final results of algorithm. Inside a smaller dataset it is possible due to the

size; here, the number of clusters are not reported since the execution do not reduce the number of k.

For a highlight of the improvement in term of the Silhouette gained through the appliance of the algorithm, we reported in Figure 6.10 an histogram of the increase of Silhouette from the value returned after $k - Means$. It is also present a bar chart which denotes the time required for both the computation of $extendeKMeans$ and its improved version. In the table we highlights in green the results returned from the algorithm during a normal execution, without exploiting every k.

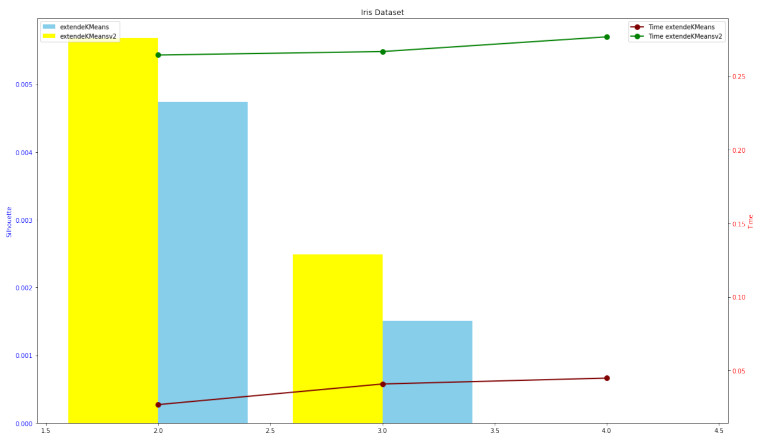

Figure 6.10: Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Iris Dataset

Table 6.9 reports the results obtained after the execution of the algorithms on Breast Cancer Dataset. The results retrieved are very similar to the Iris Dataset. In fact we obtain the best silhouete value with $k = 2$. While in the previous dataset, we get best Silhouette only into $extendeKmeansv2$, here we detect same Silhouette metric in each algorithm, due to the value reported after $k - Means$, which no technique was able to increase.
Focusing on the bigger value of K we notice a tendency of better value of the Silhouette for $extendeKMeansv2$ algorithm, as we notice into Iris Dataset. Even if the time required for the execution of the Iris Dataset was just of a few milliseconds, here $extendeKMeans$ reports a long value for the termination. For instance its execution on $k = 3$ requires more than 15 minutes for the completion.

We report into Figure 6.11 an increment in terms of the Silhouette from the common base

# of clusters	2	3	4	5	6	7
K-Means	0.4650	0.3850	0.3951	0.3745	0.3809	0.3093
Time K-Means	0.0160	0.0270	0.300	0.0400	0.0380	0.0410
extendeKMeans	0.4650	0.3886	0.3952	0.3750	0.3830	0.3120
Time extende.	0.1469	1010.7223	0.05194	0.0630	31.2905	0.0649
extendeKMeansv2	0.4650	0.3862	0.3954	0.3761	0.3824	0.3230
Time extendeV2	2.0281	2.0111	1.9648	2.0562	2.0757	2.0022

Table 6.9: Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Breast Cancer Dataset

of the Silhouette achieved into $k - Means$ algorithm. Time is dominated from the long execution required for $extendeKMeans$ with 3 clusters.

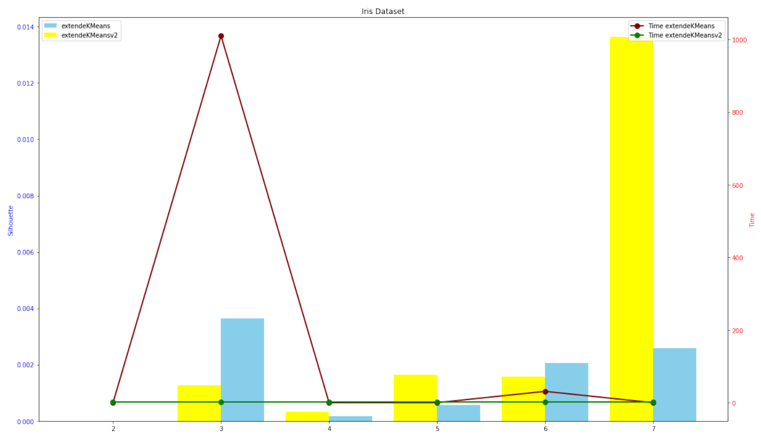

Figure 6.11: Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Breast Cancer Dataset

In Table 6.10 we summarize the results obtained of the appliance of the algorithms on the Banknote Dataset, the biggest dataset with more than 1000 samples.
It is the only case in which we obtain better results on $extendeKmeans$ compared with $extnedeKMeansv2$. In this scenario the number of samples stored into M are sensibly bigger into $extendeKMeans$ as reported from the time required for the computation on $extendeKmeans$. In fact, with $k = 3$ it requires more than 30 hours for the termination while it demands less than 1 minute into $extendeKMeansv2$. The Silhouette retrieved is the same of $extendeKMeans$. It means that even if we compared and tested more combinations, we still would not be able to detect an improvement.

# of clusters	2	3	4
K-Means	0.4336	0.3730	0.3115
Time K-Means	0.0300	0.0410	0.0631
extendeKMeans	0.4355	0.3731	0.3142
Time extende.	30182.3853	116711.7857	3538.8048
extendeKMeansv2	0.4342	0.3731	0.3142
Time extendeV2	57.4705	55.5922	57.8121

Table 6.10: Results of silhouette and time required after execution of $KMeans$, $extendeKMeans$ and $extendeKMeansv2$ on Banknote Dataset

Finally we report the improvement of the Silhouette of $extnedeKMeans$ and $extendeKMeansv2$ compared with $k - Means$ algorithm. We notice that time is dominated from $extendeKMeans$.

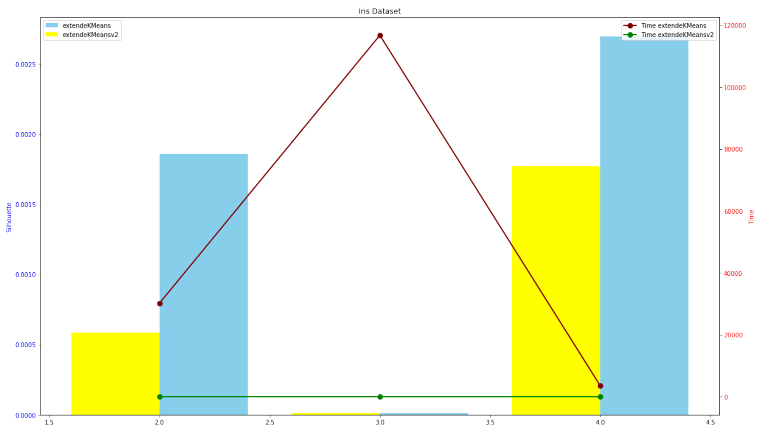

Figure 6.12: Histogram of increment of Silhouette value obtained for $extendeKMeans$ and $extendeKMeansv2$ algorithm related to $k - Means$ results. Line refers to time required in seconds for termination of the algorithm. Algorithms are applied on Breast Cancer Dataset

6.6 PAMSILHOUETTE

Here we focus on the results obtained after the execution of $PAMSILHOUETTE$ Algorithm on the Dataset proposed. First we execute the algorithm with a reduced version of the dataset, in order to measure the effectiveness of $PAMSILHOUETTE$, comparing results with $bruteForce$ algorithm.

6.6.1 Reduced Dataset

$PAMSILHOUETTE$ is executed on all of the three reduced versions of the dataset. The results are shown into Table 6.12.

# of Clusters	Iris*	Breast Cancer*	Banknote*
2	0.6791	0.7397	0.7388
3	0.5562	0.7844	0.6038
4		0.7163	
5		0.7170	
6		0.6852	
7		0.6231	

Table 6.11: Silhouette value obtained after application of $PAMSILHOUETTE$ algorithm on reduced version of Dataset

This algorithm returns into normal circumstances only on the value highlighted into the table. In order to print all the results it prints, on each execution, the different Silhouette obtained on the specific number of clusters. The dataset proposed different terminations, in fact on the Iris* and Banknote* dataset, we stop the execution after 2 iterations due to the activation of the stopping condition since we detect a fast decrease of the best value of the Silhouette. The breast Cancer* dataset presents instead a more stable value of silhouette, for this reason the algorithm is executed until $k = 7$.
$PAMSILHOUTTE$ returns for each k and the dataset analyzes the best result in term of Silhouette, as proved comparing Table 6.11 with Table 6.3, showing a great effectiveness of the given algorithm.
In this scenario we can define the result for the specific number of clusters, without the problems found into $extendeKMeans$ of reduction of k. Time results are not proposed due to the lower size of the reduced version of the dataset.

6.6.2 Complete Dataset

In this section we discuss about the results obtained from $PAMSILHOUETTE$ on the whole dataset.
First we highlight the value returned of Silhouette of each number of clusters. In Table 6.12 is highlighted the best value for each dataset. They would be the results of the algorithm.

It is important to notice the sensible improvement on the Silhouette from this approach compared with $extendeKMeans$. For instance, into Banknote dataset, we achieve an increase of Silhouette from 0.3 to over 0.5. All the complete results are available in Table 6.12 On the other hand it is highlighted in Table 6.13 the time for the completion of the algorithm,

# of Clusters	Iris	Breast Cancer	Banknote
2	0.6867	0.4649	0.5185
3	0.5553	0.4221	0.5136
4		0.4009	0.3663
5		0.3987	
6		0.3980	
7		0.3845	

Table 6.12: Silhouette value obtained after application of $PAMSILHOUETTE$ algorithm on complete version of Dataset

in average bigger than the previous approach.

# of Clusters	Iris	Breast Cancer	Banknote
2	0.9811	27.5618	416.7992
3	4.8610	99.9507	1194.1723
4		198.6331	2132.3103
5		301.1513	
6		506.1462	
7		1116.0927	

Table 6.13: Time in seconds required for application of $PAMSILHOUETTE$ algorithm on complete version of Dataset

We notice that into the Iris Dataset and Banknote dataset, the execution is stopped just after a few iterations for a rapid degradation of the value of Silhouette. Since this decrease is lower into the Breast Cancer dataset, the algorithm is executed for more iterations.

6.7 extendePAM

6.7.1 Reduced Dataset

In conclusion we analyze the experimental results of the Silhouette value and time required from $extendePAM$ algortihm and its improved version. The algorithm is very similar to the $extendeKMeans$, so it suffers on the same issue.

The number of k shown in the following results are due to the threshold set that provides a stopping condition on the algorithm. As we can notice from Table 6.17 - 6.18 - 6.19, we introduce a column related to the number of clusters into the final results returned from the algorithm. This is due to re computation of the clusters into set M. As it happens into $extendeKMeans$, if all the samples into a cluster are stored into M, it is possible that all the points are reassigned to a different cluster, reducing the number of final k. This event occurs rarely into the large dataset, but here we discuss the results achieved with just 13 samples. For this reason, only for the reduced dataset, we introduce the modified stopping condition, reported in the equation 6.1.
Here we analyze the value returned on the Silhouette after the appliance of the reduced version of the dataset. These values can be compared $bruteForce$ algorithm, in order to evaluate the effectiveness of this approach.

# of initial clusters	PAM	extende PAM	# of final clusters	extende PAMv2	# of final clusters
2	0.6791	0.6791	2	0.6791	2
3	0.5520	0.5520	3	0.5520	3

Table 6.14: Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Iris Dataset

The table 6.14 reports results obtained after the application of the algorithms on reduced Iris Dataset, comparing this results with Table 6.3 we can notice that the best value of the Silhouette is achieved with $k = 2$. Results are reported up to $k = 3$ due to the stopping condition presented into the algorithm, but we can notice that with a bigger k the algorithm is not able to gain the full results reported from $bruteForce$.
A main issue of this technique is related to the improvement from the PAM Algorithm. In fact, $extendePAM$ and $extendePAMv2$, aim to increase the silhouette value obtained after the application of PAM algorithm, but they fail their purpose since they do not provide any improvement after the testing of all the combinations into set M.

The reduced Breast Cancer Dataset suffers of the problem depicted at the beginning of this section, the reduction of the number of clusters into the results reported from the algorithm. In fact we notice that with k=5, we obtain into $extendePAMv2$ a clustering with only 4 clusters.
The results printed from Table 6.18 highlight an effectiveness of the algorithm for a smaller

# of initial clusters	PAM	extende PAM	# of final clusters	extende PAMv2	# of final clusters
2	0.7397	0.7397	2	0.7397	2
3	0.7844	0.7844	3	0.7844	3
4	0.7163	0.7163	4	0.7163	4
5	0.5233	0.5233	5	0.7163	4

Table 6.15: Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Breast Cancer Dataset

value of k. The Silhouette value reported from 2 up to 4 numbers of clusters shows that it performs the optimum clustering, while $extendePAM$ and $extendePAMv2$ fail into k = 5.
As we can evaluate on Table 6.18, a common problem of this technique is related to the real improvement from PAM algorithm. In fact in all the cases tested we are not able to improve the original value, showing for each dataset same results returned from PAM.

# of initial clusters	PAM	extende PAM	# of final clusters	extende PAMv2	# of final clusters
2	0.7388	0.7388	2	0.7388	2
3	0.5180	0.5180	3	0.5180	3

Table 6.16: Evaluation of Silhouette value obtained after execution of $extendePAM$ and its improved version on reduced Banknote Dataset

The last dataset analyzed is the Banknote dataset. As the previous cases, here we got the best Silhouetted when k = 2, but it performs poorly than $bruteForce$ on k = 3. The table shows only the first two results due to the fast decrease of the metrics, forcing stopping condition to the termination of the algorithm after the second iteration.

6.7.2 Complete Dataset

We focus here on the experimental results obtained from the execution of $extendePAM$ and $extendePAMv2$ on the entire version of the Dataset. The table 6.17 refers to Iris Dataset.

The analysis made on the entire version of the dataset highlights that best result of the Silhouette is obtained with k = 2. As into $extendeKMeans$ sections we obtain the best

# of Clusters	2	3
PAM	0.6858	0.5412
Time PAM	0.2617	0.2198
extendePAM	0.6858	0.5412
Time extende.	0.2697	0.2328
extendePAMv2	0.6868	0.5412
Time extendeV2	0.5065	0.4785

Table 6.17: Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Iris Dataset

result on the appliance of the improved algorithm. While $extendePAMv2$ gains an improvement of the original value reported after appliance of PAM Algorithm for $k = 2$, we obtain a flatting of the Silhouette index for $k = 3$, highlighting the lack of effectiveness of the algorithm, failing its purpose of the improving original Silhouette value. Due to the small size of the dataset, the time required for the termination of the algorithm is negligible.

# of Clusters	2	3
PAM	0.4645	0.3028
Time PAM	3.4065	2.8791
extendePAM	0.4645	0.3028
Time extende.	3.4475	5.4464
extendePAMv2	0.4645	0.3028
Time extendeV2	2.9200	4.8391

Table 6.18: Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Breast Cancer Dataset

Table 6.18 shows the results obtain on the Breast Cancer Dataset. In this scenario we obtain that the algorithm it is not able to detect the best Silhouette for $k = 2$, while it produces clustering with smaller Silhouette than $extendeKMeans$ and $PAMSILHOUETTE$. This behaviour happens for each k tested.
As denoted into the Iris Dataset and in all the reduced datasets, we obtain for $extendePAM$ and $extendePAMv2$ the same value of Silhouette returned from PAM algorithm. Same behaviour occurs in Table 6.19 of BankNote Dataset. It is a crucial problems the effectiveness of such algorithm.

In Table 6.19 we summarize the results obtained on the appliance of the algorithms with the Banknote Dataset. It is the biggest dataset, with more than 1000 samples. It shows the best result in terms of Silhouette for $k = 2$, obtaining a similar result of $extendeKmeans$.

# of Clusters	2	3
PAM	0.4367	0.3680
Time PAM	18.8457	18.3293
extendePAM	0.4367	0.3680
Time extende.	48.2436	938.2057
extendePAMv2	0.4367	0.3680
Time extendeV2	77.1900	75.5238

Table 6.19: Results of silhouette and time in seconds required after execution of PAM, $extendePAM$ and $extendePAMv2$ on Banknote Dataset

The value obtained after the appliance of $PAMSILHOUETTE$ sets an upper bound to the maximum Silhouette index gained from this dataset, in fact it reports a result sensibly greater than $extendePAM$ algorithm. Even for a smaller k we obtain clustering with a reduced value of the Silhouette index than the previous algorithms. As described above, it suffers of the lack of improvement from PAM algorithm. Furthermore, the time required for the termination of the algorithm starts increasing significantly. For instance, the appliance of $extendePAM$ with $k = 3$ requires more than 15 minutes.

7
Conclusion

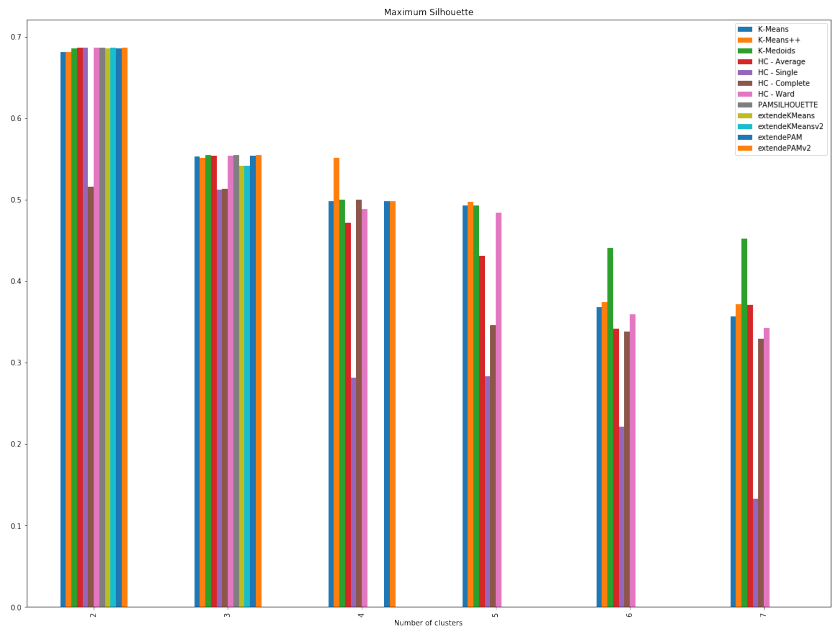

Figure 7.1: Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Iris Dataset

Silhouette is a metrics and it was crucial for the path followed into this thesis, it is posed as a cost function to be maximized, and for this reason each result was evaluated referring to the Silhouette. Since we perform the evaluation of each single algorithm, here we provide a general point of view of the different algorithms, in order to highlight the algorithms with best execute the goal of this thesis, which can be found at chapter 4. We report here the histograms comparing all the techniques exploited into this treatment and the main clustering algorithm. Figure 7.1, 7.2 and 7.3 would like to be a summary of all the results obtained into this thesis. Here we can notice how the $PAMSILHOUTTE$ algorithm performs a boost, increasing a lot the maximum value obtained compared with all the other strategies. We can also notice how the stopping condition forces a termination of the algorithm after a few iterations, but it does not influence the best results, achieved in all the cases with k is equal to 2. The most relevant results are obtained into the Banknote dataset, since it constitutes the biggest dataset proposed, with more than 1000 instances. The results are conditioned from the asymptotic complexity of the algorithm, analyzed individually for each algorithm. The main issue is related to the time elapsed for the analysis. While all the main famous algorithms require just a few seconds, even into the Banknote dataset, the algorithms presented into chapter 5 require hours for the termination, providing the results with a very small increase of the Silhouette index.

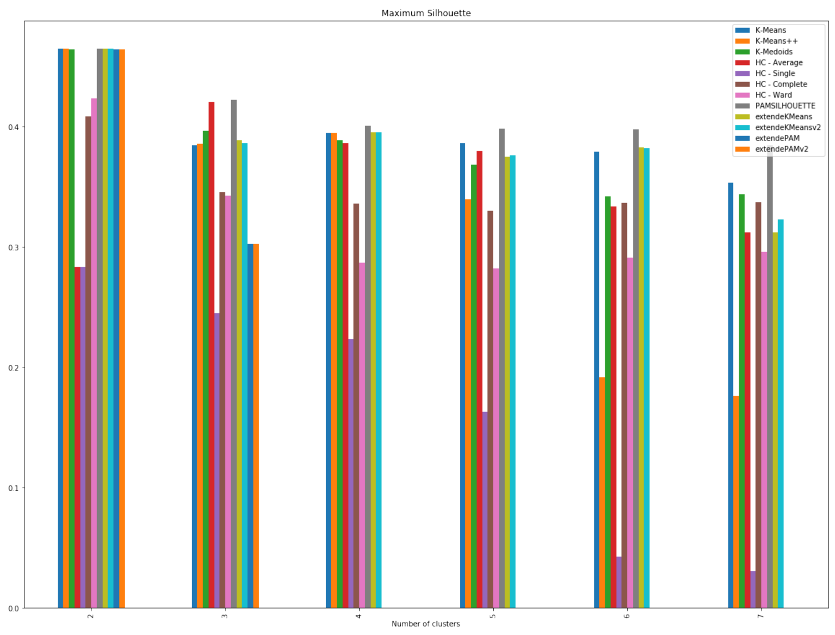

Figure 7.2: Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Breast Cancer Dataset

This results, produced here, are unbalanced compared to the amount of time required and the improvement of the silhouette; nevertheless, they are focused on detecting the best cluster.

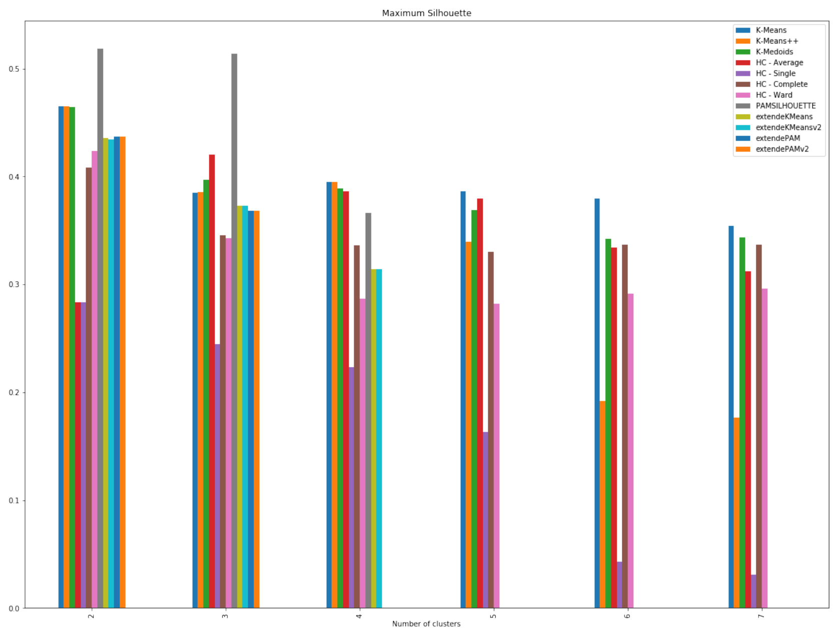

Figure 7.3: Comparison of the maximum value of Silhouette achieved from all the algorithm involved into this thesis on Banknote Dataset

The final algorithm, $extendePAM$ and its improvement are presented in order to mixing the strategies $extendeKMeans$ and $PAMSILHOUETTE$.
It starts from PAMalgorithm, and tries to improve the clustering forcing the re assignment of the points with lower silhouette. Even if this approach shows some relent results into $exntedeKMeans$, it loses all the effectiveness on the $extendePAM$, showing almost the same results gained after the execution of PAM algorithm. The first two techniques can be considered as an improvement in the Silhouette, and in particular the $extendeKMeasn$ can be considered a valid trade-off among time and Silhouette.
As future work, it could be performed a redefinition of the threshold λ_c, in order to contain a reasonable amount of points into set M, presented into $extendeKMeans$ and $extendePAM$. Another future improvement can be related to the number of clusters, in fact into this thesis we adopt a naive technique to decide the stopping condition of the algorithm. The idea suffers of the possible improvement of silhouette, after an initial decrease. The goal of the thesis is partially achieved, since some heuristic tools obtained produce an increment of the Silhouette from the traditional algorithm.

References

[1] A. Rajaraman and J. D. Ullman, *Mining of massive datasets.* Cambridge: Cambridge University Press, 2012. [Online]. Available: http://www.amazon.de/Mining-Massive-Datasets-Anand-Rajaraman/dp/1107015359/ref=sr_1_1?ie=UTF8&qid=1350890245&sr=8-1

[2] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *JSTOR: Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[3] Wikipedia contributors, "K-medoids — Wikipedia, the free encyclopedia," 2019, [Online; accessed 31-August-2019]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-medoids&oldid=909957705

[4] ——, "Elbow method (clustering) — Wikipedia, the free encyclopedia," 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Elbow_method_(clustering)&oldid=898053415

[5] ——, "Iris flower data set — Wikipedia, the free encyclopedia," 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Iris_flower_data_set&oldid=906180779

[6] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander, "Density-based clustering validation." in *SDM*, M. J. Zaki, Z. Obradovic, P.-N. Tan, A. Banerjee, C. Kamath, and S. Parthasarathy, Eds. SIAM, 2014, pp. 839–847. [Online]. Available: http://dblp.uni-trier.de/db/conf/sdm/sdm2014.html#MoulaviJCZS14

[7] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?" *J. Classif.*, vol. 31, no. 3, pp. 274–295, Oct. 2014. [Online]. Available: http://dx.doi.org/10.1007/s00357-014-9161-z

[8] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2-3, pp. 107–145, Dec. 2001. [Online]. Available: https://doi.org/10.1023/A:1012801612483

[9] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[10] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[11] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," p. 405–416, 1987.

[12] ——, *Partitioning Around Medoids (Program PAM).* John Wiley & Sons, Inc., 2008, pp. 68–125. [Online]. Available: http://dx.doi.org/10.1002/9780470316801.ch2

[13] S. Hess and W. Duivesteijn, "k is the magic number – inferring the number of clusters through nonparametric concentration inequalities," 2019, cite arxiv:1907.02343. [Online]. Available: http://arxiv.org/abs/1907.02343

[14] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973. [Online]. Available: https://doi.org/10.1080/01969727308546046

[15] J. C. Dunn†, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974. [Online]. Available: https://doi.org/10.1080/01969727408546059

[16] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, Nov. 1987. [Online]. Available: http://dx.doi.org/10.1016/0377-0427(87)90125-7

[17] H. Bersini and J. Carneiro, *Artificial Immune Systems: 5th International Conference, ICARIS 2006, Oeiras, Portugal, September 4-6, 2006, Proceedings*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006. [Online]. Available: https://books.google.it/books?id=Gp9qCQAAQBAJ

[18] L. Castro, L. de Castro, and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer, 2002. [Online]. Available: https://books.google.it/books?id=aMFP7p8DtaQC

[19] E. R. Hruschka, L. N. de Castro, and R. J. G. B. Campello, "Evolutionary algorithms for clustering gene-expression data," in *Fourth IEEE International Conference on Data Mining (ICDM'04)*, Nov 2004, pp. 403–406.

[20] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander, "Density-based clustering validation." in *SDM*, M. J. Zaki, Z. Obradovic, P.-N. Tan, A. Banerjee, C. Kamath, and S. Parthasarathy, Eds. SIAM, 2014, pp. 839–847. [Online]. Available: http://dblp.uni-trier.de/db/conf/sdm/sdm2014.html#MoulaviJCZS14

[21] C. Tomasini, L. Emmendorfer, E. N. Borges, and K. Machado, "A methodology for selecting the most suitable cluster validation internal indices," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ser. SAC '16. New York, NY, USA: ACM, 2016, pp. 901–903. [Online]. Available: http://doi.acm.org/10.1145/2851613.2851885

[22] F. Wang, H.-H. Franco-Penya, J. D. Kelleher, J. Pugh, and R. J. Ross, "An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity," in *MLDM*, 2017.

[23] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions.* Addison-Wesley Professional, 2005.

[24] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, ser. SCG '06. New York, NY, USA: ACM, 2006, pp. 144–153. [Online]. Available: http://doi.acm.org/10.1145/1137856.1137880

[25] T. V. Rechkalov, "Partition around medoids clustering on the intel xeon phi many-core coprocessor," pp. 29–41. [Online]. Available: http://ceur-ws.org/Vol-1513/#paper-04

[26] 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*, vol. 12, pp. 2825–2830, 2011.

[27] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[28] 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," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078195

[29] A. Novikov, "PyClustering: Data mining library," *Journal of Open Source Software*, vol. 4, no. 36, p. 1230, apr 2019. [Online]. Available: https://doi.org/10.21105/joss.01230