Unsupervised learning

Clustering

Gerard Caravaca Ibáñez
gerard.caravaca.ibanez@estudiantat.upc.edu

8/5/2023

Contents

1 Introduction

This coursework will be based on the review of a paper on clustering. In this case I have decided to analyse the work developed by Ismkhan, H in the paper: **I-k-means-+: An iterative clustering algorithm based on an enhanced version of the k-means** [9]. In this paper the author presents an improvement of the classical K-Means [1] algorithm and compares it with the classical implementation and with the K-Means++ [4] implementation.

In order to be able to analyse the algorithm in depth ,in this work, the algorithm will be implemented from scratch. Then, the experiments of the paper will be reproduced in order to compare results. And finally some additional experiments will be introduced in order to be able to make a well-founded critique of the results.

In research, it is typical to compare the methods and algorithms proposed to similar or competing approaches to demonstrate their empirical advantages. This requires examining the state of the art for recent methods in the field and, if necessary, implementing the competing algorithms to facilitate testing and comparison with the proposed method. That said, the ultimate goal of this paper is not to prove the worth of the algorithm but rather to show a critical attitude towards the paper and to exercise a very common task in the research field.

2 Description of the work

The paper to be analysed presents an improvement of the k-means algorithm called iterative k-means minus-plus. The improvement comes because k-means algorithm is highly sensitive to the initial centers selected and ,although numerous methods have been proposed in the literature to determine appropriate initial centers, it may not always be feasible, particularly when increasing the number of clusters. To address this issue, I-k-means-+ aims to improve the solution quality by removing one cluster, dividing another, and re-clustering the data. To expedite this process, I-k-means-+ uses various methods to determine which cluster should be removed and divided, as well as to accelerate the re-clustering process.

Before going into detail on how the proposed algorithm works, it is necessary to understand some concepts about **clustering** techniques and the k-means algorithm. Clustering is a process of grouping a set of data points or objects in such a way that the points in the same group, called a cluster, are more similar to each other than to those in other groups. The goal of clustering is to identify inherent patterns and structures in the data, such as the presence of distinct subgroups or clusters. The clustering process involves selecting appropriate distance measures or similarity metrics, defining a clustering algorithm, selecting a suitable number of clusters, and evaluating the resulting clusters to determine their quality and usefulness for further analysis or decision-making. The clustering process can be applied to a wide range of domains and applications, including data mining, machine learning, image analysis, and pattern recognition, among others. To deal with such problems, many clustering algorithms like DBSCAN [2] , CURE [3] , and k-MS [7] have been proposed in the literature. However, the K-means clustering algorithm is considered one of the most significant data mining algorithms.

2.1 K-Means definition

K-means clustering is a widely used unsupervised learning algorithm that aims to partition a set of data points into k clusters based on their similarity to each other. The algorithm works by iteratively assigning each data point to the cluster whose mean (centroid) is closest to it, and then recomputing the centroids based on the new cluster assignments. This process is repeated until convergence, when the cluster assignments no longer change.

The K-means algorithm begins by randomly selecting k initial centroids from the data points, and then assigning each point to the nearest centroid. The centroids are then updated as the means of the data points assigned to each cluster. The algorithm then iteratively repeats the assignment and update steps until convergence is reached.

In the paper, the quality of the solution is evaluated by minimizing the sum of squared Euclidean distances to the mean of each group (SSEDM). The SSEDM value of a solution is calculated as:

$$SSEDM(S) = \sum_{i=1}^{k} SSEDM(S_i) \tag{1}$$

$$SSEDM(S_i) = \sum_{\forall P \in S_i} dis(P, mean(S_i))^2 \tag{2}$$

In Equation 2, dis(,) represents the Euclidean distance between two points, while mean(S_i) refers to the center of cluster S_i. Minimizing the SSEDM precisely, even for instances with two-dimensional data points, is an NP-Hard optimization problem. However, for instances with linearly separable clusters, the k-means algorithm can effectively minimize SSEDM.

While the k-means algorithm is effective, its performance in terms of SSEDM strongly relies on the initial centers that are selected. This is why k-means is said to be very sensitive to the initialization of centres. This is the most researched improvement point for this algorithm, and this is the point that the I-k-means-+ algorithm tries to improve.

2.2 Recent works for the k-means

There are two main branches of improvement that have been explored in the k-means algorithm. The first one is the improvement of the initialization of the centers, with the goal of making the algorithm less sensitive to the randomness of the first centers selection. The k-means algorithm also loses speed since the assigning stage, where each data point should be assigned to its nearest center among existing centers, demands a significant amount of computational work.

2.2.1 Initialization methods

Initialization methods propose an intelligent way of deciding which points in the dataset will be chosen as the first centres to start the algorithm. The most studied techniques in the literature are the following:

- **Maxmin**: Choosing a set of initial centroids that are clearly differentiated from one another and accurately represent the data points is the aim of this strategy. Following is how the maxmin initialization method operates:

1. Pick the data points' first centroid at random.
2. Choose the data point that is the furthest away from all the preceding centroids for each succeeding centroid. The shortest distance between each data point and the closest centroid is calculated, and the data point with the greatest of these minimal distances is chosen.
3. Continue with step 2 until k centroids have been selected.

- **K-means++**: this technique first chooses a single data point at random to serve as the first centroid. Then determines the separation between each data point and the closest pre-existing centroid for each successive centroid. The square of the distance between each data point and the closest existing centroid determines the likelihood that it will be chosen as the next centroid. As a result, this ensures that the next centroid is far from the previous centroids. This procedure is repited until k centroids have been selelcted.

- **PCA-Part**: begins by initializing a single cluster that includes all data points. It then proceeds to divide the cluster into two separate sub-clusters in each of the k-1 subsequent steps. To determine which cluster to divide in each step, the algorithm selects the cluster with the largest partial Sum of Squared Euclidean Distances Matrix (SSEDM). To divide a cluster, the algorithm constructs a hyperplane that passes through the centroid of the cluster and is orthogonal to the principal eigenvector of the covariance matrix. The hyperplane separates the cluster into two sub-clusters based on the position of the data points relative to the hyperplane.

- **Global K-means**: starts with two centers and adds a center at each time. To add the ith center, GKM considers each data point as a candidate for the ith center. To choose the ith center, the data point which leads to the minimum value of equation 3 is selected as the ith center.

$$b_n = \sum_{j=1}^{n} max(0, d^j - ||x_n - x_j||^2) \tag{3}$$

- **Modified Global K-means**: identifies the dense sections of the data by first clustering it using a density-based clustering technique, such as DBSCAN. The initial centroids are then chosen by the algorithm from the center of these dense areas. In each cluster found by the density-based clustering algorithm, it chooses the point with the highest density.

- **MinMax**: works by finding the minimum and maximum values of each feature (dimension) in the data set. It then randomly selects K data points such that the value of each feature for each selected data point falls within the range defined by the minimum and maximum values for that feature. In other words, the selected data points are distributed uniformly across the range of each feature.

2.3 The proposed algorithm: I-k-means-+

The I-k-means algorithm seeks to improve on the two aspects outlined in the previous section. On the one hand it uses an alternative for initialisation and on the other hand it applies an improvement to the original k-means to speed up cluster reallocation. It utilizes the initialization method: **useful nearest centers** proposed by Hassan Ismkhan [6] and applies the k-means algorithm to generate an initial solution. It then proceeds with an iterative process aimed at enhancing the quality of the solution by minimizing SSEDM. During each iteration, it eliminates

a cluster (minus), splits another cluster into two (plus), and applies a re-clustering algorithm using an improved version of the k-means algorithm. As a general intuition of the algorithm is shown the Figure 1. In this figure, a cluster S_i is divided into two clusters and a cluster S_j is removed, as a consequence a new solution S is produced. It is obvious that when a cluster is removed (S_j), its data points are redistributed among other suitable clusters (S_k). These are performed, simply, by changing C_j into a random data point in S_i and then applying re-clustering process.

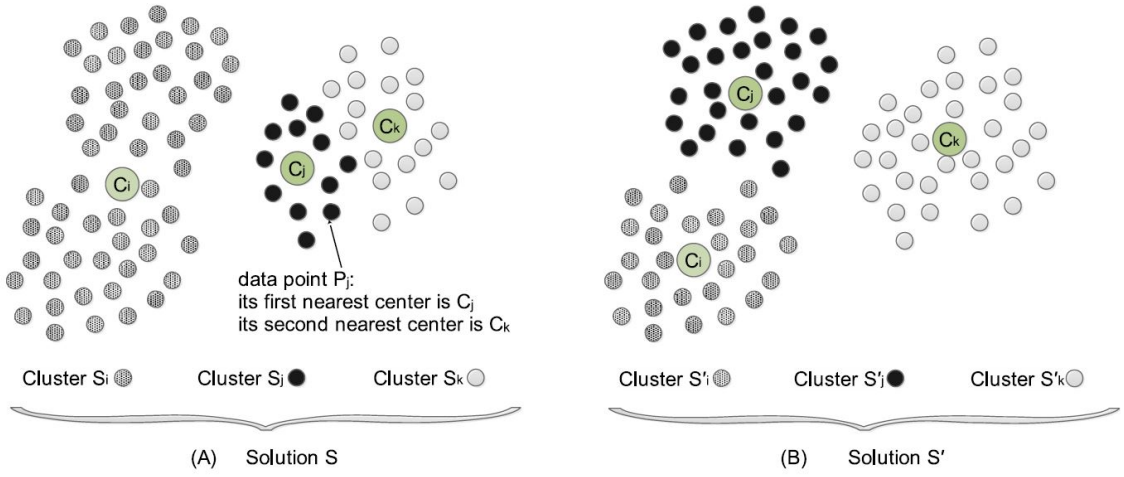

Fig. 1: Intuition of the I-k-means-+ [9].

Taking into account the process explained above, there are three main challenges to deal with. First, there should be an heuristic for finding a suitable pair of clusters such as S_i and S_j, as considering all possible pairs is not a feasible approach as it would be extremely time-consuming. Secondly,there should be an optimization of the k-means to speed up the re-clustering process when only a local portion of the solution needs to be re-clustered. Finally, we need to describe an initialization method for defining the first set of centers.

2.3.1 Initialization method

As the main aim of the k-means is to minimize SSEDM, this paper proposes a novel method. It is based on the concept of useful centers of each data point. Before stating the proposed initialization method, we need to state two simple definitions.

Definition 1 - useless.

The concept of **useless nearest center**: a center C is useless for a data point P, if for a center C_x the equation 4 is fulfilled.

$$dis(P, C_x) < dis(P, C) \wedge dis(C, C_x) < dis(P, C) \tag{4}$$

Definition 2 - useful.

The concept of **useful nearest center**: a center C is useful for a data point P, if it is not a useless nearest center of P.

The suggested initialization approach chooses the first center with the least value on the first axis, and in each subsequent iteration, the data point P with the biggest value of V_p (5) is chosen as the next center. UNC_P is the set of advantageously close centers for P. Up until the necessary number of centers are chosen, this process is repeated. This process is shown in detail in algorithm 1.

$$V_P = \frac{\frac{1}{n}\sum_{\forall c \in UNC_P, i}^{n} dis(P, c_i)}{\max_{\forall c \in UNC_P} dis(P, c)} \times \sum_{\forall c \in UNC_P} ln(dis(P, c)) \tag{5}$$

Algorithm 1 Useful nearest centers initialization

Require: Set of data points D.
Ensure: Set of centers initial centers C.

$C \leftarrow \emptyset$
$C \leftarrow C \cup D[:, 0].argmin()$
while $len(C) \neq K$ **do**
  $BEST_c \leftarrow -1$
  $BEST_v \leftarrow 0$
  for each $P \in D$ **do**
    $UNC_P \leftarrow$useful nearest centers of P
    $V_P \leftarrow V(P, UNC_P)$
    if $BEST_v < V_P$ **then**
      $BEST_c \leftarrow P$
      $BEST_v \leftarrow V_P$
    end if
  end for
  $C \leftarrow C \cup BEST_c$
end while

2.3.2 Detecting pairs method

For deciding which pair of clusters are worth to be divided and deleted it whould be estimated what is the added SSEDM by removing S_j (cost), and what is taken from SSEDM by dividing S_i (gain). Cost and Gain are defined by definitions: 3 and 4.

Definition 3 - Cost.

For a solution S, the cost of removing a cluster S_j is what is added to SSEDM(S) when S_j is removed and could be calculated by equation 6 where CCp is the second nearest center of P in solution S.

$$Cost(S_j) = SSEDM(S_j) - \sum_{\forall P \in S_j} dis(P, mean(CC_p))^2 \tag{6}$$

Definition 4 - Gain.

For a solution S, the gain of dividing a cluster S_i is what is subtracted from SSEDM(S) when S_i is divided and could be calculated by equation 7.

$$Gain(S_i) \approx \frac{3}{4} SSEDM(S_i) \tag{7}$$

Having defined the Cost() function and the Gain() function it can be defined $Cost(S_j) < Gain(S_i)$ as a heuristic to detect profitable pairs of clusters (S_i,S_j).

Apart from this definition of heuristics, we have to decide at what point the algorithm stops trying to improve the solution, as trying all combinations would not be very efficient. For this, two more heurisitics will be defined. Before stating these rules in the form of heuristics, two definitions are needed.

Definition 5 - Adjacent clusters.

A cluster S_j with center C_j is an **adjacent** of a cluster S_i with center C_i if and only if, there is a data point P in S_i (its first nearest center is C_i) such that the second nearest center of P is C_j.

Definition 6 - Strong adjacent clusters.

A cluster S_j is a **strong adjacent** to a cluster S_i if and only if S_j is an adjacent of S_i , and S_i is an adjacent of S_j.

Taking definitions 5 and 6 into account the paper defines as heuristic for delimiting the set of possible clusters for being divided or removed that if we use the pair (S_i, S_j) in a minus-plus phase then its strong adjacent clusters can not be divided or deleted in the next iterations.

2.3.3 Re-clustering method: topical k-means

The topical k-means (t-k-means) is a modified variant of the k-means algorithm that is specifically used for points that may be affected. When using the I-k-means -+ algorithm in the minus-plus phase, only a portion of the previously generated solution is altered. This is because performing a global re-clustering of the instance would be too time-consuming. Therefore, the t-k-means is applied only to the relevant points. In Figure 2, the minus-plus phase of the I-k-means-+ algorithm determines that the cluster with center C_j should be eliminated and the cluster with C_i should be split. As a result, it moves the location of C_j to a random point within the cluster centered at C_i. The t-k-means approach speeds up the re-clustering procedure by focusing on affected points and disregarding unaffected ones.

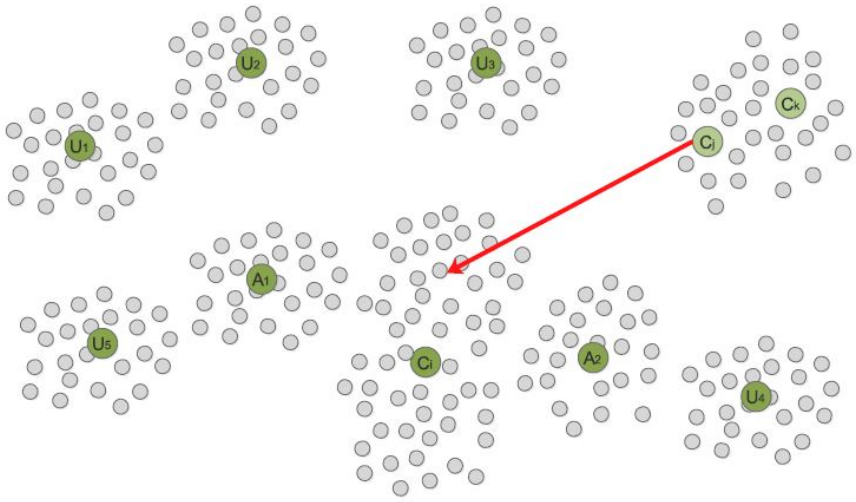

Fig. 2: Illustrative example of the tolerant k-means [9].

To apply the process described above we have to define what are the affected points in a solution. For this, we can define affected points as in definition 7.

Definition 7 - Affected points.

In a clustering solution S, a data point P is affected point of a cluster center C_i if and only if the first or the second nearest center of P is C_i.

Once all the necessary concepts have been defined, we can talk about how the algorithm itself works. The steps to implement the algorithm are shown in pseudocode 2. Taking this code and Figure 2 as a model, the algorithm can be explained as follows. determines $AC = \{C_i, C_j\}$, $AC - Adjacent = \{C_k, A_1, A_2\}$ and affected points of centers in AC and affected points of C_j before changing, during Step 2. It should be considered, some points from clusters with centers C_k , C_j , C_i , A_1 , A_2 , and U_3 are included in AP . After first execution of Step 3, not only the first and the second nearest centers of points in AP are updated, we have $PotentialAC = \{C_k, C_j, C_i\}$. This process is continued until in Step 2, AC is empty.

Algorithm 2 Tolerant K-Means

Require: current solution S, pair (C_i, C_j) from minus-plus phase and C' new center.
Ensure: new solution S' with re-clusterization applied.

$AC \leftarrow \{C_i, C_j\}$
while $AC \neq \emptyset$ **do** ▷ Step 2
    $ACAdj \leftarrow$adjacent centers of centers in AC
    $PotentialAC \leftarrow \emptyset$
    $P \leftarrow$affected points of centers in AC ▷ Step 3
    for each $P \in AP$ **do**
        update first and second nearest centers of P, considering $AC \cup ACAdj$
        if first center of P is changed from C_x to C_y **then**
            $PotentialAC \leftarrow PotentialAC \cup C_x \cup C_y$
        end if
    end for
    Update centers (mean)
    $AC \leftarrow PotentialAC$
end while

2.3.4 I-k-means-+ detailed instructions

Having defined the most important points of the algorithm, we can now focus on the clustering algorithm itself. The algorithm follows a list of 9 steps which will be described in detail below.

Step 1 of the I-K-Means-+ algorithm applies the K-Means algorithm to a given dataset using the initialization method. This produces an initial solution S with k clusters, denoted as C_i. In step 2, a variable success is set to zero. This variable keeps track of the number of successful cluster splits or merges. Step 3 involves selecting a cluster C_i with the largest value of Gain among all clusters that satisfy certain conditions. Gain is a measure of the improvement in the objective function of the clustering problem when a particular cluster is divided. If there are no clusters that meet the conditions or have a larger gain than C_i, the algorithm terminates. In step 4, the algorithm checks whether there are k/2 clusters with a larger gain than C_i. If there are, then further dividing C_i is not reasonable, and the algorithm terminates. Step 5 involves selecting a cluster C_j with the smallest value of Cost among all clusters that satisfy certain conditions. Cost is a measure of the cost of splitting a particular cluster. If there are no clusters that meet the conditions or have a smaller cost than C_j, the algorithm terminates. In step 6, C_i is divided into two clusters, C_i1 and C_i2, using a method that maximizes the gain. In step 7, the K-Means algorithm is applied to the new set of clusters, resulting in a new solution S'. Step 8 checks whether the new solution S' is better than the previous solution S in terms of the objective function. If it is, then S is updated to S', and the success variable is incremented by 1. Step 9 checks whether success is greater than k/2. If it is, the algorithm terminates. Finally, steps 3-9 are repeated until no more clusters can be removed or divided. This sequence of steps is described in pseudo-code format in the algorithm 3.

Algorithm 3 I-k-means-+

Require: Set of data points D, k.
Ensure: Clustering solution S. ▷ Step 1

$S \leftarrow$ empty clustering solution
$S \leftarrow$ useful nearest initialization
$S \leftarrow KMeans(k, init = S.centroids).fit(D)$ ▷ Step 2
$succes \leftarrow 0$
$indivisibleClusters \leftarrow \emptyset$
$unmatchablePairs \leftarrow \emptyset$
$irremovalClusters \leftarrow \emptyset$
while $True$ **do** ▷ Step 3
    if $len(indivisibleClusters) = k$ **then** Break
    end if
    $C_i \leftarrow$ center C not indivisible in S with maximum Gain ▷ Step 4
    if there are k/2 clusters having a gain larger than C_i **then** Break
    end if ▷ Step 5
    $candidates \leftarrow \emptyset$
    for each $C_j \in S.centroids$ **do**
        if $C_j \neq C_i \wedge Cost(C_J) < Gain(C_i) \wedge (C_i, C_j) \notin unmatchablePairs \wedge notadjacent(C_i, C_j) \wedge C_j \notin irremovableClusters$ **then**
            $candidates \leftarrow candidates \cup C_j$
        end if
    end for
    if $len(candidates) = 0$ **then** Break
    end if
    $C_j \leftarrow$ center C in candidates with minimum Cost ▷ Step 6
    if there are k/2 clusters having a cost lower than C_j **then**
        $indivisibleClusters \leftarrow indivisibleCluster \cup C_i$
    end if ▷ Step 7
    $newCentroid \leftarrow$ random point in cluster C_i
    $S' \leftarrow$ new solution changing center of cluster C_j by $newCentroid$
    Update S' by applying the tolerant-k-means ▷ Step 8
    if $SSEDM(S') > SSEDM(S)$ **then**
        $unmatchablePairs \leftarrow unmatchablePairs \cup (C_i, C_j)$
    else
        $irremovalClusters \leftarrow irremovalClusters \cup C_i \cup C_j$
        $indivisibleClusters \leftarrow indivisibleClusters \cup$ strong adjacents of C_j in S
        $S \leftarrow S'$
        $irremovalClusters \leftarrow irremovalClusters \cup$ strong adjacents of C_j and C_i in S
        $succes \leftarrow succes + 1$
    end if ▷ Step 9
    if $succes > \frac{k}{2}$ **then** Break
    end if
end while

3 Experimentation

In this section I will present the experiments carried out based on the implementation of the algorithm explained above. These experiments will contain both the reproduction of the experiments originally performed in the paper as well as some additional experiments to calibrate the algorithm and to be able to evaluate the work done.

The first type of datasets consists of two-dimensional synthetic problems [8], which are useful for visualizing the accuracy of algorithms. The location of the centers can indicate the quality of an algorithm's solution. On the other hand, the second and third types of datasets are real-world problems. The datasets of the second type are accessible through the UCI machine learning repository [5]. A significant bioinformatics protein dataset called KDDCUP04Bio is within the third category of datasets used in the paper. In this review I will not analyse the algorithm on the latter dataset as it is particularly recursive intensive and I believe that the other datasets provide sufficient information for the study. The complete table of datasets that were utilized in this study are listed in Table 1.

Type	**Dataset**	**Dimension**	**#instances**	**#clusters**
Synthetic	A-series A1	2	3000	20
	A2	2	5250	35
	A3	2	7500	50
	S-series S1	2	5000	15
	S2	2	5000	15
	S3	2	5000	15
	S4	2	5000	15
	Birch1	2	100,000	100
Real	Iris	4	150	3
	Human-Activity-Recognition (HAR)	561	10,299	6
	ISOLET	617	7797	26
	Letter-Recognition (LR)	16	20,000	9
	Musk	168	6598	2
	Statlog (Shuttle)	9	58,000	7
3	KDDCUP04Bio	74	145,751	2000

Tab. 1: Datasets used in the original paper's experiments

In all the experiments, two alternative algorithms, k-means (KM) and k-means++ (KM++), were used as rivals in the trials to evaluate the efficacy of the proposed I-k-means+ (IKM+). For evaluating the results, the accuracy is measured using the SSEDM. The maximum partial SSEDMs are also presented for each approach.

3.1 Experiment 1: Algorithm configuration

In this experiment, different configurations of the algorithm will be tested to calibrate it. The results will be compared with those obtained in the paper. With this, the aim is to find the most similar configuration to the one used in the paper. For this purpose, the datasets A1, A2, A3 shall be used. These contain synthetic data that facilitate the visual presentation of the results of the different clustering algorithms.

In a first test, the algorithm shall be tested as described above, i.e. using the initialization described by the author. With this, I ran 10 iterations of three clustering algorithms: K-Means,

K-Means++ and I-K-Means-+. From these 10 iterations I have presented the best iteration of both K-Means and K-Means++ and the worst iteration of K-Means-+. In this way the algorithm can be compared in the worst case. The results of this first experiment are set out in Table 2.

	Maximum of partial SSEDMs			**SSEDM**			**Runtime (s)**		
	KM	KM++	IKM-+	KM	KM++	IKM-+	KM	KM++	IKM-+
A1	1,13E+10	4,60E+09	8,45E+09	3,05E+10	1,81E+10	3,55E+10	1,00E-01	1,00E-01	0,25
A2	3,64E+09	3,36E+09	6,52E+09	2,78E+10	2,26E+10	4,34E+10	9,80E-02	1,20E-01	3,32
A3	5,76E+09	3,84E+09	1,53E+10	5,47E+10	3,45E+10	8,36E+10	1,10E-01	0,13	8,62

Tab. 2: Results of 10 runs on A synthetic datasets, using useful nearest centers initialization

As can be seen in the table, the worst results of the algorithm proposed by the paper are far from those obtained with K-Means, both in SSEDM and in runtime. To better analyse what might be going on, the results have been visualised in Figure 3, Figure 4, Figure 5. These graphs show that clearly the IKM-+ algorithm is not working as it should. The fact that the execution time is different from the other algorithms is understandable because the other algorithms may be implemented more efficiently since the scikit-learn [11] implementations have been used. In contrast, the difference in SSEDM results can only be explained by poor initialisation.

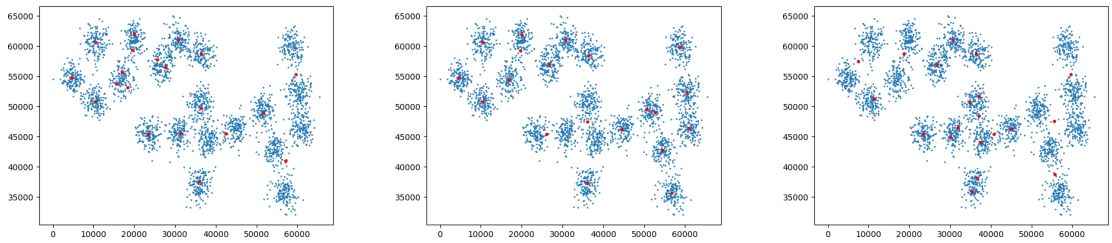

Fig. 3: For A1, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+

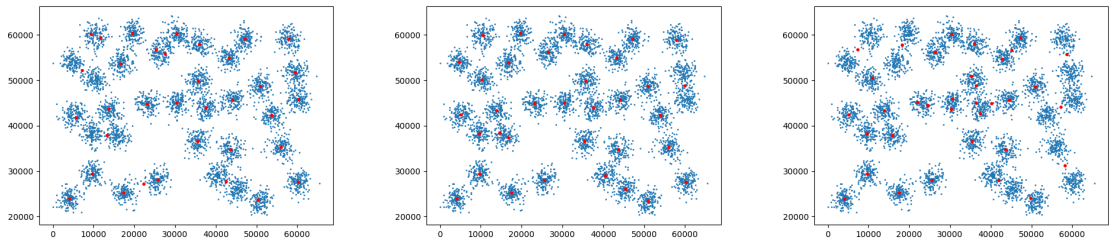

Fig. 4: For A2, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+

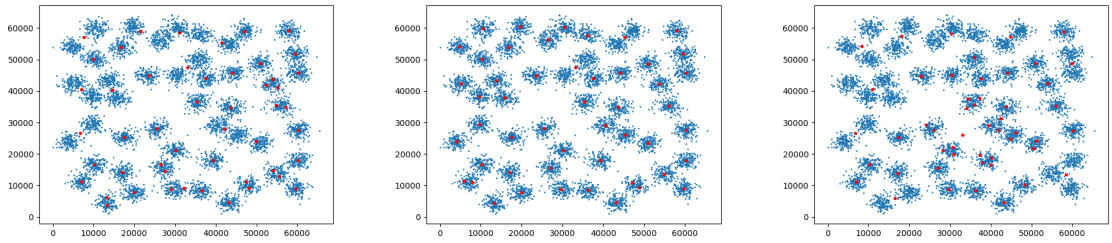

Fig. 5: For A3, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+

To verify that the problem was an initialisation problem, it was decided to repeat the experiment again, but using k-means++ as initialization. With this new configuration, the IKM-+ algorithm is expected to improve the results of the other two compared algorithms. It is also expected that the results will be closer to those obtained in the original paper. The results of this variation of the experiment are shown in tabular form in Table 3, and in graphical form in Figure 6. As expected, the IKM-+ algorithm with this configuration performs better in all cases than the two previously compared algorithms. This means that the algorithm performs well in the task of improving a first proposal but that it is still sensitive to initialization. These results are similar to those obtained in the original paper. For this reason the following experiments will be run with this algorithm configuration.

	Maximum of partial SSEDMs			**SSEDM**			**Runtime (s)**		
	KM	KM++	IKM-+	KM	KM++	IKM-+	KM	KM++	IKM-+
A1	1,13E+10	4,60E+09	3,98E+09	3,05E+10	1,81E+10	1,56E+10	1,00E-01	1,00E-01	0,08
A2	2,22E+03	3,36E+09	3,36E+09	2,78E+10	2,26E+10	2,26E+10	9,80E-02	1,20E-01	11,34
A3	5,76E+09	3,84E+09	3,39E+09	5,47E+10	3,45E+10	3,35E+10	1,10E-01	0,13	27,37

Tab. 3: Results of 10 runs on A synthetic datasets, using k-means++ initialization

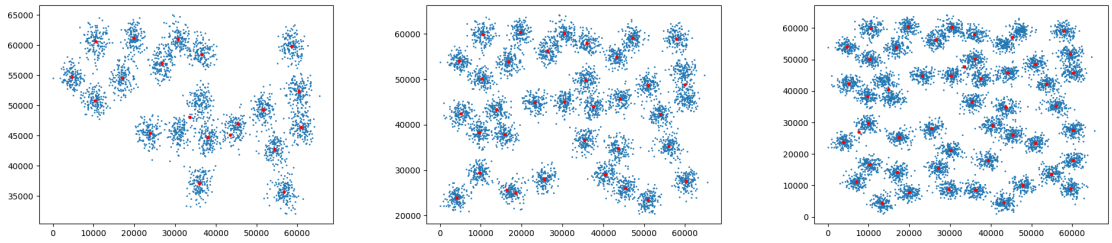

Fig. 6: Graphical results of IKM-+ in: A1 (left), A2 (middle), A3 (right).

3.2 Experiment 2: Synthetic data

In this second experiment I will apply the algorithm configuration chosen in the previous experiment to the rest of the synthetic datasets. In order to make the results comparable, I will run the experiment following the same process used above.
The results of this experiment are set out in Table 4. Some conclusions can be drawn from the results. First, K-Means++ and IKM-+ with initial centers from K-Means++ perform better

than K-Means in terms of maximum of partial SSEDMs and SSEDM. Additionaly, the worst execution of IKM-+ has lower SSEDM in some cases than KM++. As a conclusion I can confirm that in the synthesised datasets we find a very similar performance between the K-means++ algorithm and the alcorithm proposed by the paper. However, in some cases the proposed algorithm improves substantially, as in the case of the S3 dataset.

	Maximum of partial SSEDMs			**SSEDM**			**Runtime (s)**		
	KM	KM++	IKM-+	KM	KM++	IKM-+	KM	KM++	IKM-+
S1	6.61E+12	8.54E+11	8.54E+11	1.42E+13	8.91E+12	8.91E+12	0.01	0.01	2.87
S2	5.53E+12	3.81E+12	3.83E+12	1.86E+13	1.58E+13	1.59E+13	0.01	0.01	0.78
S3	4.02E+12	4.02E+12	1.62E+12	1.93E+13	1.92E+13	1.68E+13	0.02	0.02	0.94
Birch1	3.06E+12	2.89E+12	2.38E+12	1.07E+14	9.79E+13	1.02E+14	0.38	0.87	327.81

Tab. 4: Results of clustering algorithms on synthetic datasets

The conclusions drawn from the results summarised in the table above can be confirmed by looking at the graphical representation of clustering shown in: Figure 7, Figure 8, Figure 9 and Figure 10. These figures show that the centres resulting from the clustering represent more correctly the dataset in the last two algorithms used outperforming K-Means.

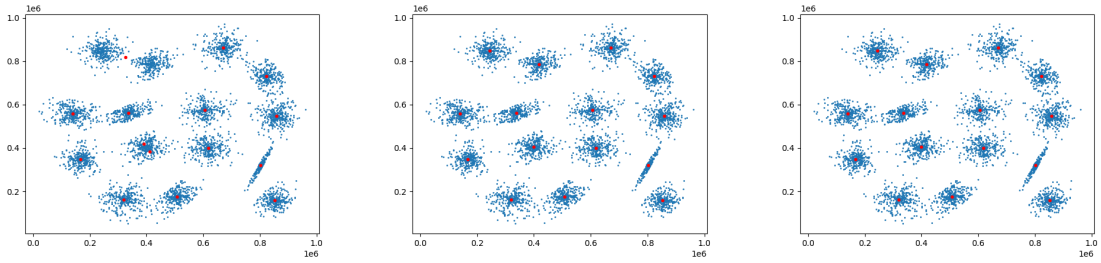

Fig. 7: For S1, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+.

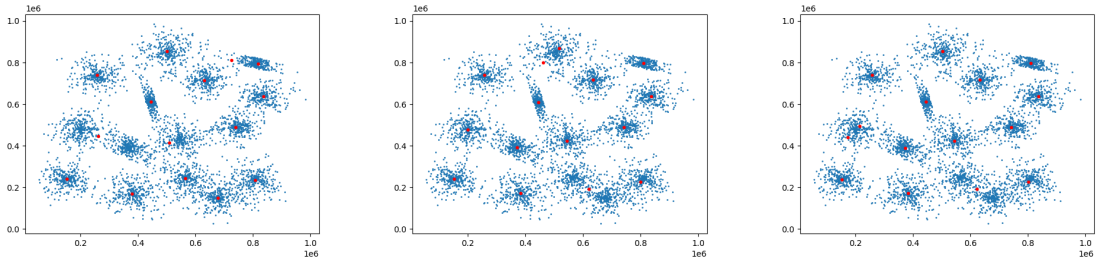

Fig. 8: For S2, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+.

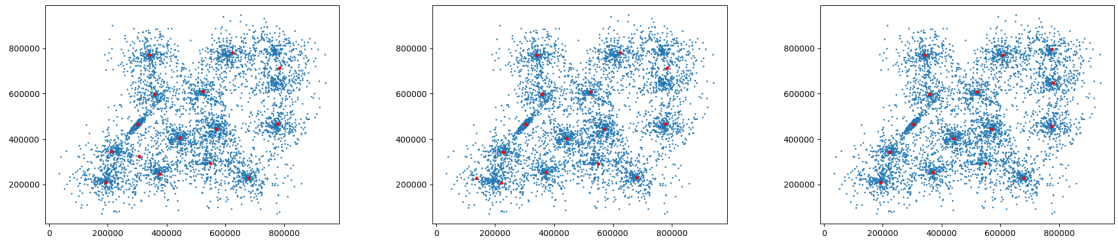

Fig. 9: For S3, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+.

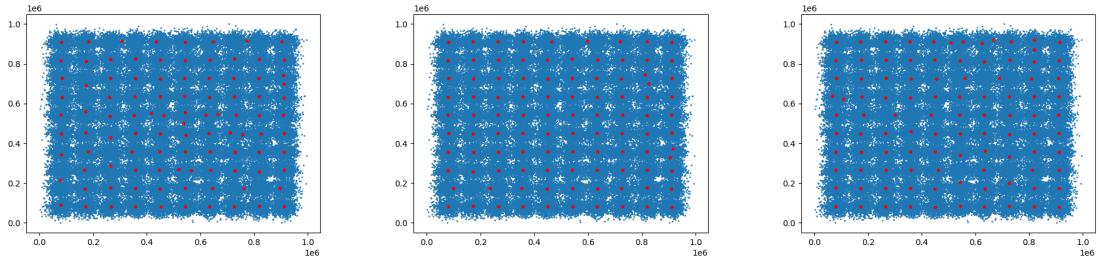

Fig. 10: For Birch1, the best solution of (left) KM, (middle) KM++, and (right) the worst solution of IKM-+.

3.3 Experiment 3: Real data

Once the algorithm has been tested on synthetic data and the results have been found to be promising. In this experiment I will analyse the application of this algorithm on real data. For this purpose, the three previously evaluated algorithms will be applied to the real datasets. The process followed in this experiment is exactly the same as in the previous experiments, the only difference being the data to which it is applied.

Table 5 shows a summary of the results obtained in this experiment. Looking at the results we can see that no conclusions can be drawn as to whether KM++ or IKM-+ is the best performing algorithm. This is because depending on the dataset one or the other works better. It should be noted that here we are showing the best result of KM++ against the worst of IKM-+ so it is normal that in some iterations there are controversial results.

	Maximum of partial SSEDMs			**SSEDM**			**Runtime (s)**		
	KM	KM++	IKM-+	KM	KM++	IKM-+	KM	KM++	IKM-+
Iris	3.98E+01	3.83E+01	3.98E+01	7.89E+01	7.89E+01	7.89E+01	1.00E-02	1.00E-02	1.00E-02
HAR	2.85E+04	2.85E+04	3.77E+04	1.31E+05	1.31E+05	1.33E+05	3.80E-01	4.50E-01	4.65E+00
Isolet	3.96E+04	3.99E+04	3.40E+04	3.81E+05	3.81E+05	3.76E+05	5.70E-01	6.00E-01	5.27E+01
LR	1.56E+05	1.33E+05	1.49E+05	8.99E+05	8.99E+05	8.98E+05	6.00E-02	8.00E-02	1.54E+01
Musk	9.04E+08	1.17E+09	1.29E+09	1.78E+09	1.19E+09	1.34E+09	6.00E-02	4.00E-02	1.00E-02
Statlog	7.79E+08	7.73E+08	1.16E+09	3.60E+09	3.57E+09	3.65E+09	7.00E-02	6.00E-02	1.71E+00

Tab. 5: Results of clustering algorithms on real-world datasets

3.4 Experiment 4: Initialization issue

As I mentioned before, to reproduce the experiments in the paper I have used the initialization of K-Means++ and not the one proposed by the author. For this reason in this experiment I will show some visual results of why the initialization proposed by the paper does not work as it should. For this I will use the synthetic datasets A1 and A2 as a sample.

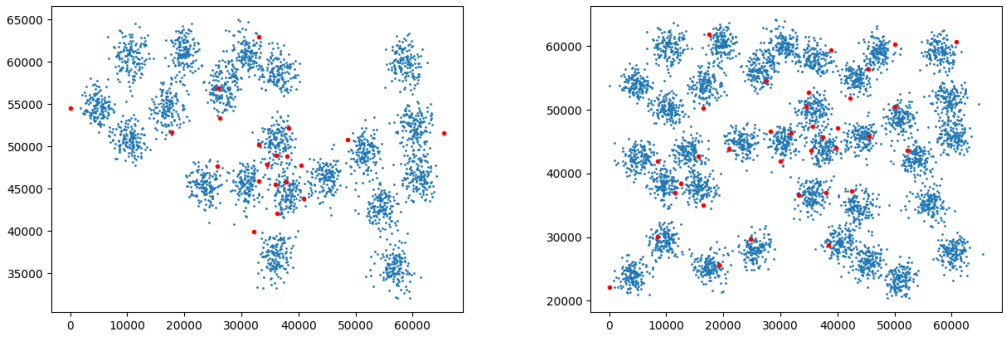

Fig. 11: Useful nearest neighbours initialization in A1 (left) and A2 (right)

As shown in Figure 11 , the initialization proposed by the paper seems to be less than ideal for the task. This is because it clusters most of the points in the centre of the data and therefore does not particularly help the clustering algorithm to separate correctly. This is why in previous experiments better results were obtained using K-Means++ as the initialization method.

4 Critical view

The paper reviewed presents an interesting improvement over the K-Means clustering algorithm. The authors explain the development of the algorithm and compare it with both K-Means and K-Means++. These experiments are carried out on both synthetic and real data. While the paper makes some compelling arguments, there are several areas where the authors could improve their analysis. While the paper makes some compelling experiments, there are several areas where the authors could improve their analysis.

In the theoretical explanation of the algorithm, the author gives an extensive and detailed explanation of both the initialization and the background of the algorithm itself. In addition, he accompanies the explanations with images and tables, which helps the reader to understand the main idea of the algorithm. However, when it comes to explaining the steps to implement the method, no pseudocode in the expected format is provided. In contrast, each step is explained in textual format, which makes it difficult to implement in code.

In the section on experiments and results, the paper presents interesting experiments that show the value of the algorithm over other well-known clustering algorithms. An important point against this aspect is that the author does not show any example of the separate initialization as I have done in experiment number 4. This has made it quite difficult to test this initialization in the development.

One final point to criticize in the paper is that the author does not share his algorithm implementation. This would have been very useful to verify the differences between their results and those obtained in this review.

5 Conclusions

To summarize, in this work I have reviewed a paper in the field of clustering. In order to carry out this review, I have made a theoretical explanation of the main concepts introduced by the paper in question. Then, I have explained step by step and with pseudocode in detail how to implement the algorithm, which complements the information provided by the author. Subsequently, I have reproduced the experiments from the paper and additionally added some experiments to validate my view of the results. Finally, after all the analysis I have given my critical point of view on the paper, highlighting the strong points and the things that I think could be improved.

When implementing the algorithm proposed in the paper, very similar results have been achieved in terms of performance. However, in terms of execution times, the results differ. This may be due to the machine on which the experiments are run or to the fact that the author uses C++ as a language and this is usually faster on average than Python. This point could be a reason for further research and improvement of this review.

The main conclusion of this paper is that the analysis of papers particularly in the field of Artificial Intelligence (AI) can be a significant challenge due to the complexity and rapidly evolving nature of the field. In this case, the paper is from 2008, which means that the proposed algorithms have been improved to date by more complex and efficient ones. Moreover, this particular paper does not make it easy to reproduce, as the step-by-step explanation of the algorithm is not detailed and the implementation has not been shared. These are the main contribution to my personal study that this work has proved to me.

6 Appendix

This last section will explain in detail how both algorithm have been implemented and the steps to follow in order to run the experiments.

6.1 Directory structure

The implementation directory follows the structure shown in Figure 12. The files shown in the figure contain the following information:

- data: contains the files for each of the datasets used.
- results: this directory was created to store tablesof results and plots.
- src: this directory contains all the classes that implement the algorithms and each of the experiments.
  - Utils: contains extra functionalities that help to read datasets and write results.
  - Proposal: implements a single solution of the clustering problem.
  - Tolerant K-Means: implements the tolerant k-means algorithm.
  - I-K-Means-+: implements the main algorithm of the paper.
- Main: is the main file to be run to start the experiments.
- Venv: contains the files that make up the execution environment.
- Requirements: This file was created to facilitate the installation of necessary libraries. Basically the libraries used are pandas, numpy and sklearn.

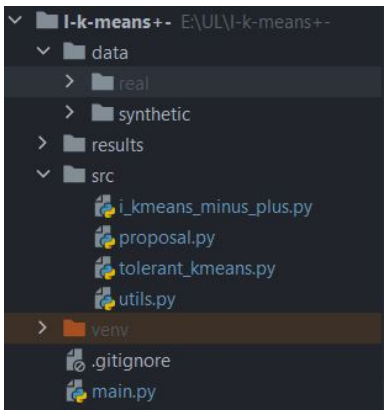

Fig. 12: Directory structure.

6.2 Python implementation

The algorithm was implemented using Python3.10 [10] and the packages pandas and numpy for loading and formatting the dataset before processing. In addition, 3 different classes have been created to implement the algorithm and another class has added to facilitate the reading of data and writing of results:

- **Proposal**: python class containing the attributes and functions needed to represent a single clustering solution. The implemented attributes are the following:
  - **Attributes**:
    - data: training data.
    - k: number of clusters.
    - centroids: centers of solution clusters.
    - nearest_centers: describes the nearest center for each instance.
    - second_nearest_centers: describes the second nearest center for each instance.
    - distance_to_center: store distances between centers and points in training set.
    - distance_inter_center: store distances between centers.
  - **Functions**:
    - useful_init(): computes the useful nearest neighbours initialization.
    - update_proposal(centers, nearest, distances): updates solution.
    - max_SSEDM(): gets maximum partial SSEDM.
    - SSEDM(): gets total SSEDM.
    - get_max_gain(): returns maximum gain between available centers.
    - k2_better_gain(gain): returns a boolean expressing if there exist more than $\frac{k}{2}$ centers with better gain.
    - get_min_cost(candidates): from the possible candidates returns the one with minimum cost.
    - k2_better_cost(): returns a boolean expressing if there exist more than $\frac{k}{2}$ centers with better cost.
    - get_random_centroid(S_i): returns a random point in the cluster S_i.
    - get_strong_adjacents(C_j): returns a list of the strong adjacent points to C_j.
    - get_affected_points(center): returs the set of affected point by center.
    - update_centroid(id, c): updates a single centroid in the list.
    - update_first_second_nearest_center(points, centers): updates complete clustering assignment.
    - update_centers(self, centers): updates list of centers.
    - recompute_distances(): update the distance arrays.
- **t_k_means**: python class containing the attributes and functions needed to compute a the tolerant k-means algorithm. The implemented attributes and functions are the following:
  - **Attributes:**
    - data: training data.
    - k: number of clusters.
  - **Functions**:
    - fit(S, $newC_j$, C_i, C_j): computes the clusters updating by aplying the tolerant k-means taking S as current solution, C_i as center of S_i, C_j as removed center of S_j and $newC_j$ as the new center of S_j.

- **I_Kmeans_minus_plus**: python class containing the attributes and functions needed to compute a the I-K-Means-+ algorithm. The implemented attributes and functions are the following:
  - **Attributes:**
    - data: training data.
    - k: number of clusters.
    - tkmeans: tolerant k-means class.
    - S: current solution.
    - init: initialization method.
  - **Functions**:
    - fit(S, $newC_j$, C_i, C_j): computes the clustering algorithm explained in this report.
    - predict(X): returns the cluster assignment for a new data X taking the fitted solution.
- **Utils**: python file containing extra functionalities for reading data. The implemented tools are:
  - **Functions**:
    - read(filename): read a dataset file from the data directory.
    - plot_data(data, centroids, dir): generates a plot representing data and cluster centers.

6.3 Execution instructions

In this section, we will go over the steps for running the code. Although the project was built using Python 3.10, it should also work with any python3 version. You can find all the necessary dependencies in the requirements.txt file located in the root directory of the project. The following execution steps has been tested both in Windows and Linux terminal:

1. Open the project folder in the terminal.
2. Create virtual environment:

```
$ python3 -m venv venv/
$ source venv/bin/activate
$ pip3 install -r requirements.txt
```

   - If you do not have the virtual environment package installed, you can install it with:

```
$ sudo apt-get install python3.8-venv
```

3. Execute an experiment:

```
$ python3 main.py -d DATA_TYPE -s RESULT_FOLDER -i INITIALIZATION_METHOD
```

   (a) The parameters of this command are the following:

   i. DATA_TYPE: refers to the type of data to be used in the experiment, being options: *real* and *synthetic*.

ii. INITIALIZATION_METHOD: refers to initialization used for starting the clustering. Options covered are: *useful*, *k-means++* and *random*.

iii. RESULT_FOLDER: refers to the path where the results will be saved. In the experiments carried out, this parameter is always *results/*.

(b) An example of this is:

```
$ python main.py -d real -s results/ -i k-means++
```

4. After executing this command all the configurations will be tested in the algorithm and dataset selected. Following, the metrics extracted from each run will be saved in an .xlsx file format. Finally, the terminal will display a summary of the complete experiment run, which you can save in a .txt file by adding *> results.txt* to the end of the above command.

References

[1] J. B. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* 1.14 (1967), pp. 281–297.

[2] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[3] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "CURE: An efficient clustering algorithm for large databases". In: *ACM Sigmod record* 27.2 (1998), pp. 73–84.

[4] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), pp. 1027–1035.

[5] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[6] Hassan Ismkhan. "An initialization method for the k-means using the concept of useful nearest centers". In: *CoRR* abs/1705.03613 (2017). arXiv: 1705.03613. URL: http://arxiv.org/abs/1705.03613.

[7] Érick Oliveira Rodrigues et al. "k-MS: A novel clustering algorithm based on morphological reconstruction". In: *Pattern Recognition* 66 (2017), pp. 392–403. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2016.12.027. URL: https://www.sciencedirect.com/science/article/pii/S0031320316304484.

[8] Pasi Fränti and Sami Sieranoja. *K-means properties on six clustering benchmark datasets*. 2018. URL: http://cs.uef.fi/sipu/datasets/.

[9] Hassan Ismkhan. "I-k-means-+: An iterative clustering algorithm based on an enhanced version of the k-means". In: *Pattern Recognition* 79 (2018), pp. 402–413. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2018.02.015. URL: https://www.sciencedirect.com/science/article/pii/S0031320318300694.

[10] Python Software Foundation. *Python Release Python 3.10.0*. 2021. URL: https://www.python.org/downloads/release/python-3100/.

[11] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. Accessed: 2023-04-06. 2021. URL: https://scikit-learn.org/stable/index.html.