“CLUSTERING ALGORITHM SELECTION BY META-LEARNING”

Panagiotopoulos Georgios

Prof. Doulkeridis Christos

Submitted in partial fulfillment of the requirements for the Diploma degree of

“Information Systems & Services”

Postgraduate Program “Big Data and Analytics”

Department of Digital Systems

University of Piraeus

Athens, February 2020

Keywords

Data Characterization, Clustering, Meta-Learning, Algorithm Ranking, Algorithm Selection, Meta-Knowledge

Abstract

Data clustering attempts to classify a database into object groups based on the similarities between the objects in question. The quest for a good-quality solution can become a complex process because of its unsupervised existence. There is currently a wide range of clustering algorithms, and it can be a slow and expensive process to select the best one for a given problem. For every dataset that is related to clustering problems, there is an exhaustive procedure that requests from a Data Scientist firstly to test each clustering algorithm to find the most suitable one. A system that recommends the clustering algorithm and guides the user for selecting the right one would be a great tool that would provide significant benefits to the scientific community. Rice formulated the Algorithm Selection Problem (ASP) in 1976, which postulates that the output of the algorithm can be predicted based on the structural features of the problem. Meta-learning has been used successfully for recommendation tasks with algorithms. It uses machine learning to induce meta-models capable of predicting the best algorithm of a new dataset. Experimental results show that the recommendation improves with these meta-attributes. With a significant accuracy, it is presented that a system could indeed recommend a clustering algorithm for an "unknown" dataset only by examining its meta-attributes firstly. Also, this Master Thesis discusses the relevance to the recommendation of each meta-feature.

Περίληψη

Η συσταδοποιήση δεδομένων είναι μια προσπάθεια ομαδοποίσης μιας βάσης δεδομένων σε ομάδες αντικειμένων βασισμένες στις ομοιότητες των εν λόγω αντικειμένων. Η αναζήτηση μιας ποιοτικής λύσης μπορεί να γίνει μια περίπλοκη διαδικασία λόγω απουσίας της επιτήρησης. Αυτή τη στιγμή υπάρχει ένα ευρύ φάσμα αλγορίθμων ομαδοποίησης και μπορεί να είναι μια αργή και δαπανηρή διαδικασία για την επιλογή του καλύτερου για ένα δεδομένο πρόβλημα. Για κάθε σύνολο δεδομένων που σχετίζεται με προβλήματα ομαδοποίησης, υπάρχει μια εξαντλητική διαδικασία που ζητά από έναν Data Scientist πρώτα να ελέγξει κάθε αλγόριθμο ομαδοποίησης για να βρει το πιο κατάλληλο. Ένα σύστημα που συνιστά τον αλγόριθμο ομαδοποίησης και καθοδηγεί τον χρήστη για την επιλογή του σωστού θα ήταν ένα εξαιρετικό εργαλείο που θα προσέφερε σημαντικά οφέλη στην επιστημονική κοινότητα. Ο Ράις διατύπωσε το πρόβλημα επιλογής αλγορίθμου (ASP) το 1976, το οποίο υποθέτει ότι η παραγωγή του αλγορίθμου μπορεί να προβλεφθεί με βάση τα δομικά χαρακτηριστικά του προβλήματος. Η μετα-μάθηση έχει χρησιμοποιηθεί με επιτυχία για εργασίες συστάσεων με αλγόριθμους. Χρησιμοποιεί την εκμάθηση μηχανών για να προκαλέσει μετα-μοντέλα ικανά να προβλέψουν τον καλύτερο αλγόριθμο ενός νέου συνόλου δεδομένων. Τα πειραματικά αποτελέσματα δείχνουν ότι η σύσταση βελτιώνεται με αυτά τα μετα-χαρακτηριστικά. Με σημαντική ακρίβεια, παρουσιάζεται ότι ένα σύστημα θα μπορούσε πράγματι να συστήσει έναν αλγόριθμο ομαδοποίησης για ένα "άγνωστο" σύνολο δεδομένων μόνο εξετάζοντας πρώτα τα μετα-χαρακτηριστικά του. Επίσης, αυτή η Διπλωματική εξετάζει τη συνάφεια με τη σύσταση κάθε μετα-χαρακτηριστικού.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work took nearly a whole year to complete, and during that time, a lot of things happened in my life that made working on this thesis more difficult than I had expected. I want to take some time here and thank the people who made this job possible, given everything that happened.

It goes without saying that I am grateful to my supervisor, professor Doulkeridis, for his tremendous help and patience to accomplish this work. He was always open and compassionate for any question that I had. His persistence pushed me to accomplish what we had discussed from the beginning.

Since my father's loss ten years now, there is one person in my life that has been stood to me in every decision I made, right or wrong, and this person could not be other than my mother. I want to thank her for her love to me and my brothers, and her support to make all our choices possible.

Finally, I would like to thank all my co-workers at Angelicoussis Group who firstly trust me and gave me the opportunity, five years now, to work with them in this high-level company.

Chapter 1: Introduction

One of the main Machine Learning (ML) applications is data clustering [1]. With the growing interest in understanding, processing, and summarizing data automatically, clustering algorithms have been successfully applied to various application domains, such as anomaly detection, gene expression analysis, community detection, and object segmentation.

To obtain a model with a good predictive or descriptive efficiency, the selection of an appropriate algorithm to address a given ML task is fundamental. Each algorithm attempts to model and solve a problem by extracting information on those characteristics, leading researchers to investigate a large number of algorithms. The selection of the most suitable algorithms among this large number is typically based on empirical observation or previous experiences of the user, which can be subjective and have a high computational cost

Problem-solving is a central and well-defined concept of Computer Science in general. After formulating a computational problem, an algorithm is used to solve the problem. Algorithms are formal and methodical approaches to a certain problem that returns a solution within finite (and reasonable) time and space. A problem may be (and is often) approached by more than one algorithms. Much research has been invested in analyzing the properties of algorithms and their efficiency in problem-solving, to be able to understand the impact of choosing a particular algorithm. Unfortunately, there is no single algorithm that achieves the best performance overall instances of a problem class. Rice [2] formulated the *Algorithm Selection Problem* (ASP), which proposes that there is a relation between the characteristics of a problem and has been tackled in different research fields, well known as *meta-attributes*.

1.1 META - LEARNING

The *meta-attributes* compose the knowledge (*meta-knowledge*) to select the most suitable algorithm for a new, unseen problem, and this can be achieved by a system that aims to learn which problems characteristics contribute to a better

performance of one algorithm over the others. This is a *meta-learning* system that deals with the ASP by learning about the behavior of the learning algorithms. The meta-knowledge, also known as meta-data, consists of the meta-attributes and the meta-target. The meta-attributes are features or characteristics derived from the problems. The meta-target is the meta-learning device destination vector. A comprehensive review of meta-learning and ASP can be found in Smith-Miles [3][4], in which the author described how their application is applied to various fields of research, such as time series prediction [5], sorting [6], and optimization [3][4], among others.

1.2 WHAT THIS WORK IS ABOUT

In this Master Thesis, it is proposed the direction of selecting dynamic algorithms, using divergent algorithms on different sub-instances of the original problem, in a way that ends up making use of the best aspects of each available algorithm, by applying the most suitable algorithm to each sub-instance to yield the minimum possible cost. In other words, during the problem-solving process, we leave usable algorithms in between. Of course, how this dynamic selection works must be general and not unique to a specific problem or set of algorithms. Otherwise, the reusability benefit of this work would be limited if any

Work is focused on clustering Algorithms, where it is designed and implemented a new framework in which each “unknown” dataset will be imported, and the most suitable clustering algorithm will be recommended in order then to be run. With this framework, it is reduced a significant time from configuring and running all suitable clustering algorithms and then choose the most relevant one.

The way that this “system” recommends a clustering algorithm comes from extracting dataset’s meta-data and then compares its results with a pool of already known clustering datasets. Then the system recommends the most suitable clustering algorithm according to its similarity of their meta-data. The whole meta-data idea is based on previous work by Daniel Gomes Ferrari and Leandro Nunes de Castro [7]. In their work, are proposed two different ways to collect the meta-data. In the first way, meta-attributes are based on the object’s attributes. The other way proposes meta-attributes based on the distance between the objects of the clustering problem instead of on the object attributes themselves.

A series of 50 datasets (Appendix A), four algorithms, and five internal indices are used to validate the proposals, and the meta-knowledge is constructed with two distinct sets of meta-attributes: the conventional approach, the modern unsupervised distance-based method. In performance evaluation, three hybrid ranking approaches are used: an existing system using the average rank level and two new methods focused on score and competition. Next, this information is applied to a meta-learning system to learn the relationship between the features of the problems and the output of the algorithms, and a selection process tests the quality of meta-knowledge.

1.3 HOW IT IS DONE

The implementation uses 50 datasets that are collected from different sources, extracts their meta-data, runs clustering algorithms, and after store their meta-attributes and their performance in a dataset. A framework was built, in which each dataset where imported, its meta-data were extracted, and then all four clustering algorithms were run. Using the five internal indices, the most efficient one was selected and stored along with the dataset's meta-data. The dataset that contains the meta-data and the most suitable clustering algorithm is used for training and test purposes, where a classification algorithm *K-NN* [8] is run. K-NN is an instance-based algorithm that classifies an object based on a search for its nearest neighbors.

1.4 OVERALL PICTURE

The algorithm recommendation systems are fascinating research topics, as also, they can be precious to the research community. Overall, the results of the below work were encouraging, and they prove that the meta-learning system can be an extremely positive addition in the clustering algorithm recommendation.

1.5 THESIS OUTLINE

This Thesis is organized as follows: In Chapter 2 it will be provided the necessary theoretical background, which includes clustering algorithms, internal indices, and a basic introduction to the K-NN classification algorithm. In Chapter 3 it

will be provided a more detailed problem statement and an overview of the related work in the area. In Chapter 4 it will be described the approach to extract the appropriate meta-knowledge. In Chapter 5 it will be presented all the implementation details of this approach. In Chapter 6 it will be provided the results of this work and finally, in Chapter 7 it will be discussed the final results and the comparison between the two meta-data approaches and will also be suggested future work.

Chapter 2: Background Knowledge

This chapter provides the background information required for further comprehension of the research discussed in this study. We explain the notion of machine learning and its principal categories first; we then delve into the subarea of Algorithm Selection, explaining tis goals and methodologies. Finally, we introduce the evaluators of the clustering algorithms known also as *clustering indices*

2.1 MACHINE LEARNING

Machine learning [9][10] is an interdisciplinary research field which combines ideas developed in the fields of computer science, mathematics, statistics, operational research, cognitive science and engineering to provide machine intelligence. Machine learning is a research field which is dedicated to automated system learning.

Automated learning systems range in complexity and application. They may be defined by a very simple memorization-based learning system [11] such as the one filtering unwanted emails (spams) based on memorized table of unwanted senders, to those using inductive reasoning in dynamic environments to perform more complex tasks [12]. Machine learning has recently become the tool of choice when it is important to extract useful information from huge, complex datasets. In this section we will shed some light on the field of machine learning with a focus on the various approaches available.

There are many different ways to describe machine learning. It can be defined by the learning process or the model (engine) that it uses to determine. Another way of classifying a learning system is based on the principle of learning that occurs, another is based on the level of interaction that a learning system employs with its feedback. An interactive learner will engage with his / her environment (data) by conducting an experiment, for example, to gain more input information, whereas a passive learner will simply rely on observing the input information. Machine learning research is generally divided into three distinct branches. Supervised [13], Unsupervised [14] and Learning for Reinforcement [15]

2.1.1 Supervised Learning

A desired set of outputs for given inputs (learning phase) are given to the machine in supervised machine learning. It then asks the machine to produce an output for the newly arriving input.

The optimal result could be either a class mark used for classification in machine learning systems [16], or a real number in those used for regression [17]. The optimal performance in regression problems is to predict or calculate a new value for a dependent variable using values derived from data attributes based on learning from previous training sets.

Nonetheless, the optimal output for classification problems is to classify the input data into predefined labels or groups using a training set of previously classified data.

2.1.2 Unsupervised Learning

Unlike supervised machine-learning systems, the machine is not provided with the desired result during the training phase in unmonitored machine-learning systems. It is the duty of the computer (model) to learn how to construct a function that clusters the input data according to their statistical characteristics.

This clustering is not established by any supervisory mechanism during the learning phase or by any user; rather, it is carried out on the basis of the discovered relationship between the different features of the input and the modeled clusters.

Deciding how to organize inputs into clusters that share common properties is an essential job of the unsupervised learning systems. Clustering is the process of dividing the datasets into sub-sets which share common features. The K-means clustering algorithm (will be explained in a later section) is a popular technique for unsupervised learning. Other examples of unsupervised learning are those that are used in our framework and will be presented more extensively at the following chapter (chapter 2.2), like the *particle swarm optimization* (PSO) [24] and *Density-based spatial clustering of applications with noise (*DBSCAN) [19] and *Single linkage* [20].

2.2 CLUSTERING ALGORITHMS

As previously mentioned, this Master Thesis tries to figure out a mechanism in which in an unknown dataset, a clustering algorithm will be recommended. However, it is necessary to clarify what clustering is and where it can be implemented. According to Zubin and Joseph [18], the goal of cluster analysis or clustering is to group a collection of objects in such a way that objects in the same group (called a cluster) are more similar to each other (in a sense) than objects in other groups (clusters).

Cluster analysis itself is not a particular algorithm but the overarching process that needs to be resolved. Multiple algorithms that differ significantly in their understanding of what constitutes a cluster and how to find them effectively can achieve this. Popular cluster notions include groups with small distances between members of the cluster, large areas of data space, intervals, or unique statistical distributions. Consequently, clustering can be formulated as a problem of multi-objective optimization. The required clustering algorithm and parameter settings (including parameters such as the distance function to be used, a density threshold, or the number of predicted clusters) depend on the individual data set and the planned application of the results.

Cluster analysis as an action is not an automatic task but an iterative knowledge discovery process or multi-objective interactive optimization involving trials and failures. Sometimes, preprocessing data and modeling parameters need to be changed until the output achieves the desired properties.

2.2.1 DBSCAN

DBSCAN stands for *Density-based spatial clustering of applications with noise* and is a proposed data clustering algorithm by Hans-Peter Kriegel, Martin Ester, Jorg Sander, and Xiaowei Xu in 1996 [19]. It is a non-parametric density-based clustering algorithm: given a set of points in some space, it aggregates points that are tightly packed together (points with many adjacent neighbors), labeling them as outliers lying alone in low-density regions (whose closest neighbors are too far away).

The algorithm that runs for DBSCAN has one goal and this aims to define dense regions that can be calculated by the number of the objects near a given point. DBSCAN requires two important parameters: *epsilon* ("eps"), and *minimum points*

("MinPts"). The eps parameter defines the neighborhood radius around a point x. MinPts parameter is the minimum number of neighbors within the radius of “eps”. Each point x in the dataset is marked as a core point, with a neighbor count greater than or equal to MinPts. We say x is border point if the number of its neighbors is less than MinPts, but it belongs to some core point z's epsilon-neighborhood. Ultimately, if a point is neither a center nor a boundary point, then a noise point or an outlier is named. The following figure shows the different types of points using MinPts= 4 (core, boundary, and outlier points). Here x is a core point because (neighbors epsilon(x) = 4), y is a boundary point because (neighbors epsilon(y) < MinPts), thus belongs to the core point x (epsilon) neighborhood. Z is a noise level, at last.

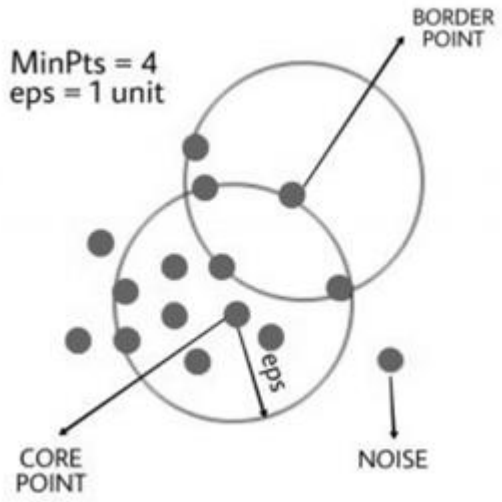

Figure 1: Visualization of the DBSCAN approach.

The algorithm of density-based clustering works as below, where this begins with an unvisited arbitrary point of departure. The epsilon-neighborhood of this point is retrieved, and if it contains enough points, a cluster is started. The argument is otherwise known as noise. Remember that this point may be found later in a sufficiently sized epsilon-environment of a different point and thus become part of a cluster. If a point is found to be a dense part of a cluster, it is also part of that cluster in its epsilon-neighborhood. Therefore, all points which are located within the neighborhood are included, as is their own neighborhood when they are dense as well. This process continues until the cluster that is related to density is completely defined. Then, a new unvisited point is retrieved and analyzed, resulting in a further cluster or noise being detected.

```
Density-based spatial clustering

DBSCAN(dataset, eps, MinPts){
# cluster index
C = 1
for each unvisited point p in dataset {
         mark p as visited
         # find neighbors
         Neighbors N = find the neighbor points of p

         if |N|>=MinPts:
             N = N U N'
             if p' is not member of any cluster:
                 add p' to cluster C
}
```

Figure 2: The DBSCAN pseudo-algorithm

2.2.2 Single-Linkage

Single-linkage clustering [20] is one of several Hierarchical clustering methods in statistics. It is based on grouping clusters in bottom-up fashion (agglomerative clustering), combining two clusters at each step that contain the closest pair of elements that are not yet part of the same cluster as one another. One downside of this approach is that it tends to produce large, thin clusters in which neighboring elements

of the same cluster have small distances, but elements at opposite ends of a cluster can be much farther from each other than two elements of other clusters. This can lead to difficulties in class definition that could usefully subdivide the data.

That participant is within a cluster of its own at the beginning of the agglomerative clustering process. The clusters are then grouped sequentially into larger clusters until all of the components end up in the same cluster. The two clusters divided by the shortest distance are combined at each stage. The definition of' shortest distance' is what distinguishes the different methods of agglomeration. In single-link clustering, the distance between two clusters is determined by a single pair of elements, namely those two elements which are closest to each other (one in each cluster). The shortest of those connections that remain at any point causes the two clusters whose elements are involved in fusing together. The process is also known as the closest clustering of neighbors. The product of the clustering can be visualized as a dendrogram showing the cluster fusion series and the distance at which each fusion occurred.

Mathematically, the linkage function – the distance D(X,Y) between the clusters X and Y – is described by the expression

$$D(X,Y) = \min_{x \in X, y \in Y} d(x,y)$$

where X and Y are any two sets of elements that are considered as clusters, and d(x,y) denotes the distance between the elements x and y.

2.2.3 K-Means

K-means clustering, as defined by James MacQueen in 1967 [21], is a method of vector quantization that is common for cluster analysis in data mining, originally from signal processing. The goal of the clustering of k-means is to divide n observations into k clusters. In that, each observation belongs to the cluster with the nearest mean, serving as a cluster prototype. These results in data space partitioning into Voronoi cells [22]. K-Means minimizes distances within the cluster (squared Euclidean distances), but not regular Euclidean distances, which would be the most challenging problem for Weber's problem: The mean optimizes square errors, while the Euclidean distances are minimized only by the geometrical median. For example, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally complicated (NP-hard) however, efficient heuristic algorithms easily converge to a local optimum. Typically these are close to the expectation-maximization algorithm for Gaussian distribution mixtures through an iterative optimization approach used by both k-means and Gaussian mixture modeling. Both use cluster centers to model the data. However, clustering k-means tends to find clusters of comparable spatial magnitude, while the method of expectation-maximization enables clusters to have different forms.

The algorithm has a loose relationship with the k-nearest neighbor classifier, a common classification machine learning technique that is often confused with k-means by the name. Applying the1-nearest neighbor classifier to the k-means-obtained cluster centers classifies new data into existing clusters. This is known as the Rocchio algorithm [23] or the nearest centroid classifier.

The K-means algorithm in data mining begins with the first group of randomly selected centroids, which are used as the starting points for each cluster, and then perform iterative (repetitive) calculations to optimize centroid positions. This prevents the formation and optimization of clusters; if either the centroids stabilized— their values are unchanged because the clustering was efficient or the number of iterations specified was achieved.

```
K-Means

Initialize k means with random values


For a given number of iterations:
    Iterate through items:
        Find the mean closest to the item
        Assign item to mean
        Update mean
```

Figure 3: The K-Means pseudo-algorithm

2.2.4 Particle Swarm Optimization

In computational science, *particle swarm optimization* (PSO) is a computational approach that optimizes a problem by attempting to iteratively develop a candidate solution with respect to a given quality measure. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles

over the position and velocity of the particle in the search space based on simple mathematical formulae. The movement of each particle is determined by its locally best-known location but is also directed towards the best-known search-space positions, which are modified as other particles find better positions. This should push the swarm towards the best solutions.

Credited initially to Kennedy, Eberhart, and Shi [24], PSO was initially intended to model social behavior as a stylized depiction of organism activity in a bird flock or fish school. The algorithm was simplified, and optimization was observed. Kennedy and Eberhart's book [25] discusses many facets of PSO and swarm intelligence in philosophical terms. Poli carries out a detailed study of PSO applications. Recently, Bonyadi and Michalewicz [26] have released a comprehensive review of theoretical and experimental work on PSO.

PSO is a metaheuristic, as it makes few or no assumptions about optimizing the problem and can search for vast spaces of candidate solutions. Metaheuristics like PSO, however, do not guarantee that an optimal solution is ever found. PSO often does not use the gradient of the problem being optimized, which means that PSO does not require that the issue of optimization be distinguished as needed by traditional methods of optimization such as gradient descent and quasi-newton methods.

A simple version of the PSO algorithm works by having a population of candidate solutions (called a swarm) (called particles). According to a few simple formulae, these particles are moved around in the search space. The motion of the particles is guided by their own best-known position in the search space, as well as the best-known position of the whole swarm. When improved positions are found, these will then come to guide the swarm's movements. The process is repeated, and it is expected that a satisfactory solution can eventually be seen but not guaranteed.

```
Particle Swarm Optimization

For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If fitness value is better than best fitness value
(pBest) in history
            set current value as new pBest
    End

    Choose the particle with best fitness value of all the
particles as the gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End

While maximum iterations or min error criteria is not attained
```

Figure 4: The PSO pseudo-algorithm

2.3 CLUSTERING INDICES

The concept of validation steps for clustering results has, therefore been a challenging issue that numerous methods have sought to overcome. A Clustering Validation Index (CVI) can be used to measure the quality of the clustering performance and multiple of them are described by M. Halkidi [27] [28]. A CVI's goal is to estimate the most suitable K based on cluster compactness and separation. The indices of validity may be divided into three categories [29]: internal, external, and relative. The external validation index takes advantage of prior knowledge, the internal index is based on data information only, and multiple clustering findings are compared in a relative CVI. Alternative approaches for calculating the number of clusters also exist, for example, by evaluating the stability of the clustering method [30].

Cluster validation takes into account the quality of the result of a clustering algorithm, trying to find the partition which best fits the nature of the data. The number of clusters given as a parameter for many clustering algorithms should be determined

based on the data's natural structure. Like the best clustering solution, also the number of clusters is not always straightforward, and there may be several' false' answers. The number can also depend on the resolution, i.e., whether the separabilities within and between the clusters are considered globally or locally.

During the implementation, the clustering validation was measured by 5 different cluster internal indices. These are the *Davies-Bouldin index, Hubert-Levin index* (also known as C-Index), *Dunn index, Calinski-Harabasz index,* and *Silhouette index*. The sections below will be given further information about these indices.

2.3.1 Davies-Bouldin index

The *Davies-Bouldin index* (DBI) was introduced by David L. Davies and Donald W. Bouldin in 1979 [31] as a metric for evaluating clustering algorithms. This is an internal assessment scheme where the analysis of how well the clustering was performed is achieved using the inherent quantities and features of the dataset. This has a downside that this approach recorded a good value that does not mean the best retrieval of the information.

How does this index work? First, as δ_k is denoted the mean distance of the points belonging to cluster C_k to their barycenter $G^{\{k\}}$:

$$\delta_k = \frac{1}{n_k} \sum_{i \in I_k} ||M_i^{\{k\}} - G^{\{k\}}||$$

Also, it is indicated by

$$\Delta_{kk'} = d(G^{\{k\}}, G^{\{k'\}}) = ||G^{\{k'\}} - G^{\{k\}}||$$

The distance between the barycenters $G^{\{k\}}$and $G^{\{k'\}}$ of clusters C_k and $C_{k'}$. One computes, for each cluster k, the maximum M_k of the quotients $\frac{\delta k + \delta k'}{\Delta kk'}$ for all indices $k' \neq k$. The Davies-Bouldin index is the mean value, among all the clusters, of the quantities M_k:

$$\mathcal{C} = \frac{1}{K} \sum_{k=1}^{K} M_k = \frac{1}{K} \sum_{k=1}^{K} \max_{k' \neq k} \left(\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}} \right)$$

2.3.2 Hubert and Levin index

The *Hubert and Levin index* is an internal clustering index discovered by L.J Hubert and J.R Levin in 1976 [32]. It is also known as *C-Index*. The *C-Index* is an

evaluation measure expressed as: $[d_w - min(d_w)] / [max(d_w) - min(d_w)]$, where d_w is the sum of all n_d within-cluster distances, $min(d_w)$ is the sum of the n_d smallest pairwise distances in the data set, and $max(d_w)$ is the sum of the n_d biggest pairwise distances. All pairwise distances in the data set must be measured and stored in order to calculate the C-Index. In this case of binary data, distance storage poses no issues because only a few distances are feasible. Calculating all distances, however, will make the index prohibitive for large data sets. The maximum value of the second difference is taken as the proposed number of clusters.

2.3.3 Dunn index

The *Dunn index* was firstly introduced by J. C. Dunn in 1974 [33] as a metric for evaluating clustering algorithms. This is part of a validity index category, including the *Davies-Bouldin index* or *Silhouette index,* as it is an internal assessment scheme where the outcome is focused on the clustered data itself.

Like all other such indices, the aim is to classify compact cluster sets with a small variance between cluster members and well separated where the means of different clusters are sufficiently far apart compared to the variance within the cluster. A higher Dunn index would indicate better clustering for a given cluster assignment. One of the drawbacks of using this is the computational cost as cluster numbers and data dimensionality increase.

In order to compute the Dunn index, it has to be denoted by d_{min} the minimal instance between points of different clusters and d_{max} the largest within-cluster distance. The distance between clusters C_k and $C_{k'}$ is measured by the distance between their closest points:

$$d_{kk'} = \min_{\substack{i \in I_k \\ j \in I_{k'}}} ||M_i^{\{k\}} - M_j^{\{k'\}}||$$

And d_{min} is the smallest of these distances $d_{kk'}$:

$$d_{min} = \min_{k \neq k'} d_{kk'}$$

For each cluster C_k, there is a D_k as the largest distance separating two distinct points in the cluster (sometimes called the diameter of the cluster):

$$D_k = \max_{\substack{i,j \in I_k \\ i \neq j}} ||M_i^{\{k\}} - M_j^{\{k\}}||.$$

Then d_{max} is the largest of these distances D_k:

$$d_{max} = \max_{1 \le k \le K} Dk$$

The Dunn Index is defined as the quotient of d_{min} and d_{max}:

$$C = \frac{dmin}{dmax}$$

2.3.4 Calinski-Harabatz index

The *Calinski-Harabatz index* discovered by T. Calinski and J. Harabasz in 1974 [34] is an evaluation index for clustering algorithms and is defined as the ratio between the inter-cluster dispersion and the intracluster dispersion.

$$(BCD * (N - k)) / (WCD * (k - 1))$$

where n is the number of data points, and k is the number of the clusters. The minimum value of the second difference is taken as the proposed number of the clusters.

2.3.5 Silhouette index

The *Silhouette index* is also an internal index responsible for the interpretation and validation of consistency with clusters of data. The technique offers a brief graphical depiction of how well each item was categorized. It was firstly introduced by Peter J. Rousseeuw in 1987 [35].

The silhouette value is the measure of how close an entity is to its own (cohesion) cluster relative to other (separation) clusters. The outline varies from −1 to + 1, where a high value means that the object is well aligned with its own cluster and poorly matched to neighboring clusters. If most objects have high value, then the configuration for the clustering is correct. If many points have a low or negative value, then there may be too many or too few clusters in the clustering configuration. The outline can be measured with any distance metric like the distance from the Euclidean or the distance from Manhattan.

The silhouette index is calculated as below. For each point, M_i, its mean distance to each cluster. One defines the within-cluster mean distance *a(i)* as the mean distance of point M_i to the other points of the cluster it belongs to: if $M_i \in C_k$, we thus have

$$a(i) = \frac{1}{n_k - 1} \sum_{\substack{i' \in I_k \\ i' \neq i}} d(M_i, M_{i'})$$

On the other hand, while evaluating the mean distance $\partial(M_i, C_{k'})$ of M_i to the points of each of the other clusters $C_{k'}$:

$$\mathfrak{d}(M_i, C_{k'}) = \frac{1}{n_{k'}} \sum_{i' \in I_{k'}} d(M_i, M_{i'})$$

Now, *b(i)* is denoted as the smallest of these mean distances:

$$b(i) = \min_{k' \neq k} \mathfrak{d}(M_i, C_{k'})$$

The value *k'*, which realizes this minimum indicates the best choice for re-affecting, if necessary, the point M_i to another cluster than the one it currently belongs to.

For each point M_i, one then forms the quotient

$$s(i) = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)}$$

Which is called the silhouette width of the point. Its quantity takes values between -1 and 1: a value near 1 indicates that the point M_i affects clusters to the right whereas a value near -1 indicates that the point should be affected to another cluster.The mean of the silhouette widths for a given cluster C_k is called the cluster mean silhouette and is denoted as s_k:

$$\mathfrak{s}_k = \frac{1}{n_k} \sum_{i \in I_k} s(i)$$

Finally, the global silhouette index is a calculation of the mean of the mean silhouettes through all the clusters:

$$C = \frac{1}{K} \sum_{k=1}^{K} sk$$

Chapter 3: Problem Statement

3.1 ALGORITHM SELECTION: A META-PROBLEM

As stated in the introductory section, the critical issue that this work seeks to address is that of selecting algorithms for clustering problems. Selecting the right algorithm for a problem is a decision that depends on more than just the theoretical assumptions of each algorithm regarding time and memory complexity. Hardware information is significant, as is the expected distribution of input from a statistical point of view.

Inner details of an algorithm, in combination with the above, are also relevant. That makes the selection of algorithms a complex problem in itself, a kind of meta-problem. As such, it requires the same amount of exploration and analysis as any standard computational problem. Solutions should be formulated generically and methodically so that they can be applied to each instance of this meta-problem with minimal modification.

A per-instance solution for any problem from scratch is usually not cost-effective, especially when solutions that apply to the general nature of that problem have already been proposed. This is why there are generic algorithms that use to general problem formulations, such as sorting algorithms, search algorithms, algorithms for limit satisfaction, etc. Since we have developed the process of selecting the appropriate algorithm as a problem, it would be reasonable to try to create solutions for that problem that will apply in the general case rather than a method per instance, where each instance is treated as a whole new problem.

This directly implies that for each instance, the solution will not be the same, which is why we refer to this as a selection of dynamic algorithms. This means that a decision will be made based on the instance's specifics but will be made based on a set of general rules that will apply regulations that have been defined to make the best decision based on the particulars.

3.2 SELECTING AND RECOMMENDING AN ALGORITHM

Consider selecting or recommending an appropriate subset of ML algorithms for a given task. The problem can be interpreted as a search problem, where the search space includes the individual ML algorithms, and the goal is to identify the best performing learning algorithms package. Figure 5 demonstrates a general framework for the collection of learning algorithms.

The process can be divided into two phases, according to this framework. In the first phase, the objective is to identify an appropriate sub-set of learning algorithms given a training dataset using available meta-knowledge. The performance of this step is a ranked subset of ML algorithms, reflecting the new, reduced space for bias. Instead, the second phase of the process involves searching through the reduced area. To identify the best alternative, each learning algorithm is evaluated using different performance parameters (e.g., consistency, accuracy, recall, etc..).

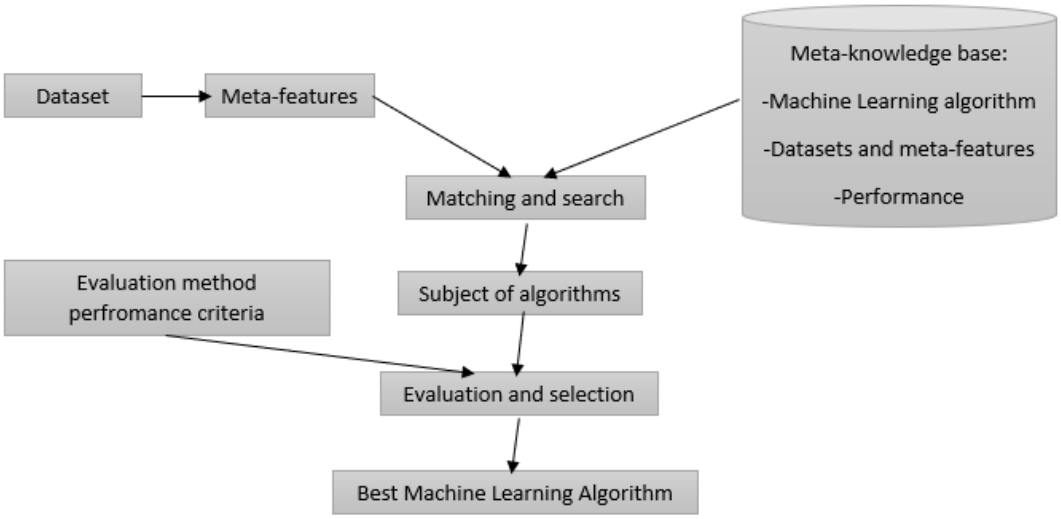

Figure 5: Selection of ML algorithms: finding a reduced space and selecting the best learning algorithm

This varies from traditional approaches by using a foundation of meta-knowledge. As described above, one important goal in meta-learning is to research how meta-knowledge can be extracted and used to benefit from past experience. Information contained in the base of metaknowledge may take various forms. It may include, for example, a set of learning algorithms showing good (a priori) performance on datasets similar to the one being analyzed; algorithms characterizing Machine

Learning algorithms and datasets and metrics available to measure similarity or task-relatedness of the dataset. Meta-knowledge, therefore, includes not only useful information for dynamic bias selection but also functions and algorithms that can be invoked to generate new useful information.

It is noted that metaknowledge usually does not remove the need for search entirely but rather offers a more efficient way of searching through alternative spaces. The success of the search process depends on the quality of the metaknowledge available.

3.3 ALGORITHM RECOMMENDATION WITH META-LEARNING

A recommendation algorithm system can be defined as a tool that supports the user in the data mining process selection algorithm step. It indicates which algorithm to use to achieve the best possible results given a dataset. If there are sufficient computational resources available to try multiple algorithms, it should also indicate which ones to execute and in which order. In practice, such a system can be said to guide the experimental process in a data mining application.

To achieve this goal, predicting the *true performance* of the algorithms accurately is not as important to an algorithm recommendation method as predicting their *relative performance* is. Therefore the algorithm recommendation task can be defined as the algorithm ranking according to their predicted performance.

It is important to use data describing the performance of algorithms and the characteristics of problems, which we will refer to as metadata to address this problem using a machine learning approach. The figure below illustrates how meta-learning with the ranking recommendation by Average Ranking works (Figure 6)

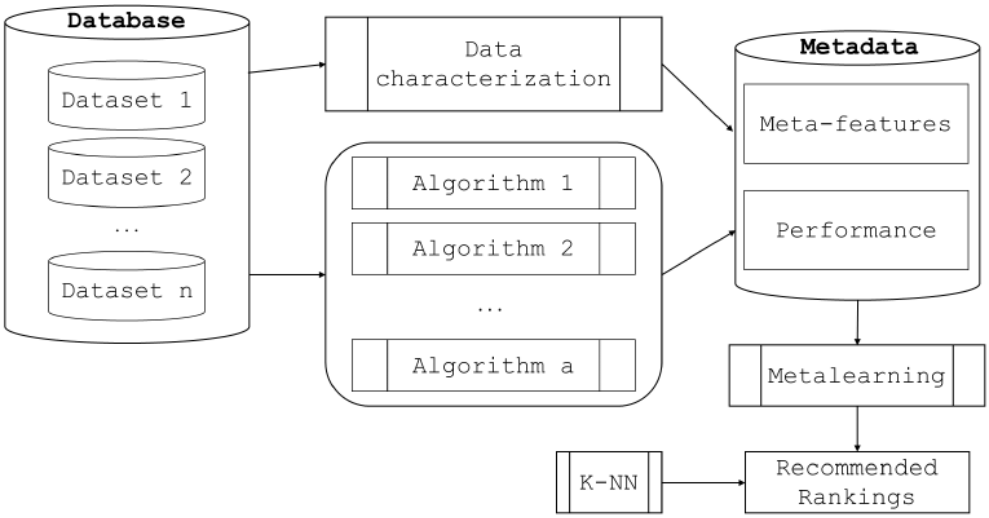

Figure 6: Meta-learning to obtain meta-knowledge for algorithm selection

The performance data is used for determining algorithm rankings. The primary function of this learning assignment is these scores, called target rankings. The criteria used to describe the problems are characteristics that are independent of the specific task.

3.4 RELATED WORK

Treating algorithm selection as a problem is not a whole new idea in itself. It was officially stated first by J.R. Rice as a computational problem in 1976 [36]. The Portfolio of Algorithms was introduced as a framework to treat algorithm collection as a formal computational problem [37].

Because algorithm selection was identified as a computational problem, a few methods have been developed as to how it should be solved. These include studying the instance to be solved and selecting a suitable algorithm for it [38], running multiple algorithms from the portfolio in parallel, and ending as soon as one solution is obtained by the fastest algorithm [39] and switching algorithms to runtime [40].

The work of Daniel Gomes Ferrari and Leandro Nunes de Castro [7] is the one closest to this Master Thesis, since the extraction of meta-features and the ranking method are based on their work. Also, the use of the specific clustering algorithms, as well as the clustering indices, are referred to this work.

Chapter 4: Proposed Approach

To provide solution for the issues of selecting the proper clustering algorithm mentioned in the previous sections, the proposed method uses two sets of meta-features. One that is related to the object's attributes and based on statistical measures such as the percentage of missing values in a dataset and the other based on the Euclidean distance between each dataset's instance.

The general idea is the creation of a system that will import any dataset and preprocess its data according to some "pre-processing rules". Then it will extract its meta-features (object/distance-based) and store it to a local database. After that, it will try to find similarities in these meta-features with an already stored collection of meta-features, and finally, it will propose a clustering algorithm according to the similarities.

To achieve the above, we created a local database that contains a collection of 50 extracted meta-features along with their preferable clustering algorithm that were obtained by 50 datasets from different domains. Below it is described our steps in creating such a system and how efficient that system is.

4.1 DATASET PRE-PROCESSING

In this approach, the experiments used datasets collected from the different locations in the world wide web such as Kaggle, University of Eastern Finland, etc. These datasets cover various domains, including engineering, biology, medicine, physics, and robotics. Thus, unlike the datasets used in the previous studies, the datasets used here cover a wider variety of application domains.

However, before starting experimenting in each one of them, some actions needed to be taken first, known as pre-processing steps. In all datasets, it was necessary to remove all information about labels or classes. Also, all nominal values were converted into numbers following the alphabetical order of the unique values. With the above two actions, the datasets contained now only numerical values. Then, if any attribute had the same value for all its objects, this attribute was removed.

The same action (attribute removal) was also applied to any attribute that had a different value for each its object. Any object that had a missing value was replaced

by zero (0), and if any attribute had more than 40% missing values, this was removed as well. As a last necessary action, after completing all the above steps, was to normalize the remaining data over the interval [0,1].

To summarize, all preprocessing steps can also be found below.

Table 1: Datasets pre-processing

Pre-Processing Datasets
(1) Information about labels or classes were removed
(2) Nominal values were converted into numbers
(3) Removal of attributes with same value at all their objects
(4) Removal of attributes with different value at all objects
(5) Removal of attributes of having more than 40% missing values
(6) Replaced any object with missing value with 0
(7) Data Normalization [0,1]

4.2 OBJECT'S ATTRIBUTE APPROACH

The object's attribute approach was proposed by D.G. Ferrari and L.N. de Castro, as already mentioned before. At their approach, data similarity can be observed by extracting nine meta-attributes based on the object's attributes of a dataset (Table 2).

Table 2: Meta-attribute set based on the attributes

Meta-attribute	***Description***
MA_1	Log2 of the number of objects
MA_2	Log2 of the number of attributes
MA_3	Percentage of discrete attributes
MA_4	Percentage of outliers
MA_5	Mean entropy of discrete attributes
MA_6	Mean concentration between discrete attributes
MA_7	Mean absolute correlation between continuous attributes
MA_8	Mean skewness of continuous attributes
MA_9	Mean kurtosis of continuous attributes

The first four meta-attributes (MA1 – MA4) extract global information about the problem. They are universal values that can be easily obtained. At the 5th meta-attributes and after the attributes are separated to discrete and continuous. To automatically distinguish whether an attribute is discrete or not so are defined the following rules. First, if the attribute has real numbers as objects, then the attributes are considered as continuous. Secondly, if the number of unique values is less than 30% then these attributes are regarded as discrete. Lastly, if the above is not occurring, then the attributes are regarded as continuous.

In order to identify the outliers of the datasets, the Interquartile Range (IQR) [41] rule is used. This method is based on the boxplot, as V. Hoge [42] suggested in 2004. Data suggests the beliefs are grouped around a certain core value. The IQR says how the "mean" values are distributed out; it can also be used to determine when some of the other values are "too far" from the central value. These "too distant" points are called "outliers" because they "lie outside" the range we expect them to be within. The IQR is the length of the box in your box-and-whisker plot. An outlier is any value that is more than 1⁄2 times the length of the box at either end of the box. The Lower Limit is calculated as the subtraction of the first quartile (Q1) to the IQR that is multiplied first by 1,5 as follows

$$\text{Lower Limit} = Q1 - (1{,}5 * IQR)$$

The Upper Limit is calculated as the addition of the third quartile (Q3) to the IQR multiplied by 1,5 as follows

$$\text{Upper Limit} = Q3 + (1{,}5 * IQR)$$

To summarize, an object is considered an outlier if at least one of its attributes has a value outside its respective upper or lower limits.

4.2.1 Extracting meta-features

In this method, each dataset is imported to our system, and then all the data cleaning rules (mentioned on 4.1) are applied. As a second step, the framework checks each dataset's column if it contains a discrete object or continuous and for it breaks the dataset to two smaller. One that contains the discrete values and the other contains the columns with continuous values.

For each column that contains discrete values (according to the rules on 4.2), the entropy and the concentration are calculated. Then, the mean value of each of these calculations is stored into two registers (MA_5 and MA_6). After that, is counted the size of the total objects of the dataset and then the length of the columns-attributes which that dataset contains. Then, we produce and store the values for the first two meta-attributes (MA_1 and MA_2). While using the IQR method, we are dividing the dataset into two quantiles (0.25 and 0.75), and then we are calculating the outliers according to the rules. In order to find the percentage of the outliers in each dataset, we divide the outliers from the number of total objectives (which we calculated before), and this value is stored in a separate register (MA_4). Knowing which attributes have discrete values and which have not, it is easy to calculate the percentage of discrete columns of the dataset (MA_3).

On the other hand, for each column with continuous attributes, it is calculated and stored (MA_7, MA_8, MA_9) the correlation, the skewness, and the kurtosis of these columns. To calculate the correlation, it is used the *Pearson* method [43], which is the covariance of the two variables separated by the sum of their standard deviations. The description type includes a "product moment" that is, the mean (first moment about the origin) of the sum of mean-adjusted random variables. The framework continues, with the creation of two tables where they will be stored the score of each clustering algorithm according to the clustering indices and the ranking between all the clustering algorithms.

For each dataset, we will execute the four clustering algorithms (DBSCAN, SL, PSO, K-Means), and they will be evaluated by the five clustering indices (DB, CH, SIL, DU, HL). The first two clustering algorithms (DBSCAN and SL) require as a configuration only the threshold. We compare the results for the number of clusters for each of them, and we are using this number of clusters as an input to the next algorithms (PSO, K-Means), which request this number as an input.

Starting with the DBSCAN, we are running this algorithm by changing its threshold in order to have the best result according to the clustering indices. After we end up in the ideal number of clusters according to the DBSCAN algorithm, we are storing this number to a register, and also we are saving the score of all clustering indices for DBSCAN.

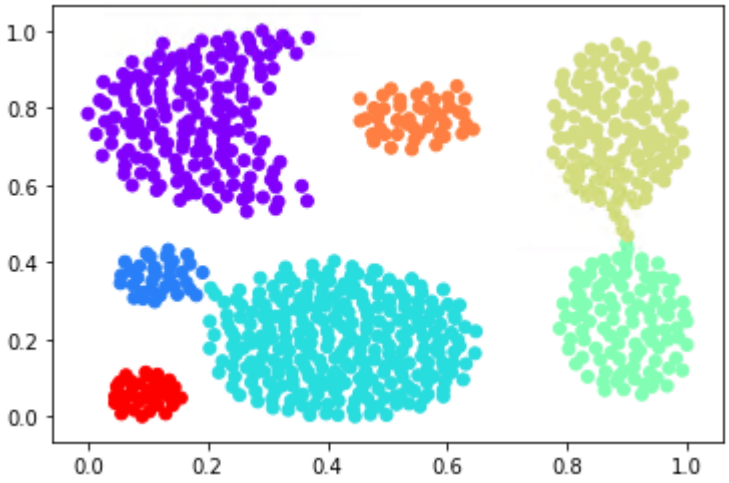

Figure 7: Plot example for DBSCAN

Then, we continue with the Single – Linkage algorithm. Following the same procedure, we are running multiple times the Single – Linkage algorithm by changing the threshold until the most efficient number of clusters is found, always according to clustering indices.

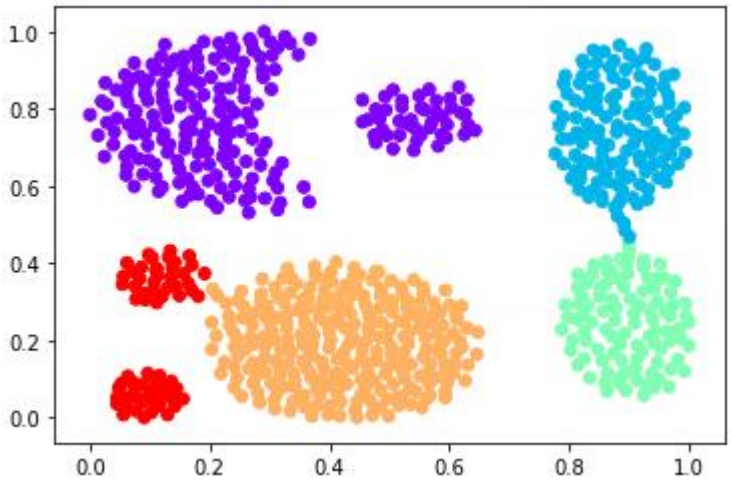

Figure 8: Plot example for Single – Linkage

We store again this number as well as the score of all five indices for this algorithm, and we compare the stored indices score between the Single – Linkage and the DBSCAN algorithm. For the above example (Figures 7, 8), according to the clustering indices, the adequate number of clusters were DBSCAN clusters, which were seven. At the below table, it can be observed the objective for each internal index we are using.

Table 3: The internal indices and their respective domains and search objectives

Internal Index	***Interval***	***Objective***
Davies-Bouldin [20]	[0;+∞)	Min
Calinski-Harabasz [23]	[0;+∞)	Max
Silhouette [24]	[-1;+1)	Max
Dunn [22]	[0;+∞)	Max
Hubert-Levin [21]	[0;+1]	Min

After comparing the indices, we take the number of clusters from the winner, and we use it as an input to the next clustering algorithm (K-Means). There, we again run a clustering algorithm, and we are storing the scores of the internal indices.

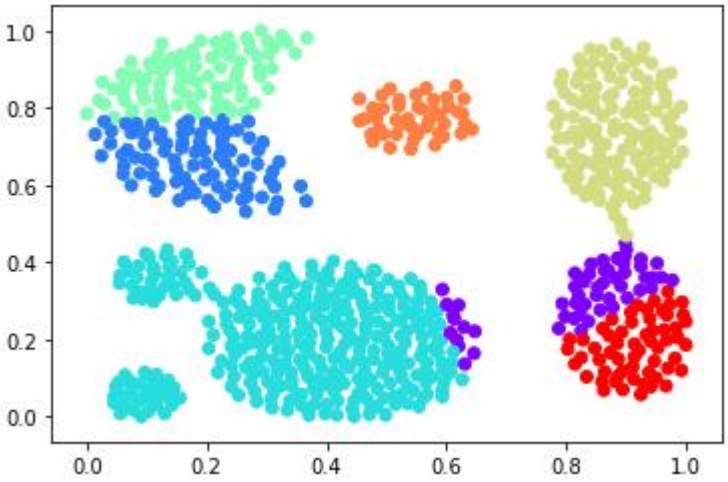

Figure 9: Plot example for K-Means

The same procedure is followed back for the next and last one clustering algorithm (PSO). We store its indices values again.

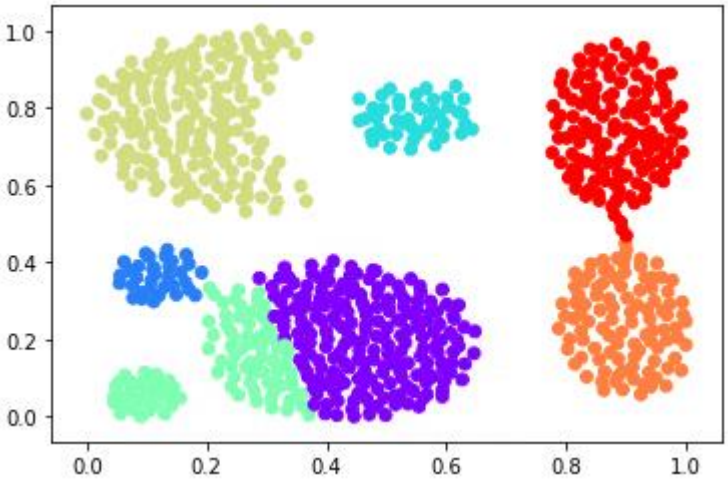

Figure 10: Plot example for PSO

Completing the execution of all four clustering algorithms for each dataset we have stored their clustering indices values.

Table 4: Example of stored clustering indices for all algorithms

	KM	SL	PSC	DBS
DB	0.816542	0.594282	0.612971	0.447282
HL	0.117292	0.158000	0.128158	0.131023
DU	0.021213	0.327290	0.039598	0.375184
CH	826.035171	1430.926201	1674.167609	1650.839885
SIL	0.482723	0.624418	0.562663	0.612571
Final Rank	0.000000	0.000000	0.000000	0.000000

There, somehow, we should consider one of them as a winning clustering algorithm. In order to find this winning algorithm, we are following the methods of *score ranking* and *winner ranking*.

Table 5: Example of point system for score and winner ranking

```
{'CH': {'DBS': 8, 'KM': 4, 'PSC': 10, 'SL': 6},
 'DB': {'DBS': 10, 'KM': 4, 'PSC': 6, 'SL': 8},
 'DU': {'DBS': 10, 'KM': 4, 'PSC': 6, 'SL': 8},
 'HL': {'DBS': 6, 'KM': 10, 'PSC': 8, 'SL': 4},
 'SIL': {'DBS': 8, 'KM': 4, 'PSC': 6, 'SL': 10}}
```

The method of score ranking is based on race tournaments, in which at the end of each race the pilots get points. At the end of the tournament, the winner is declared the pilot (algorithm), with more points. In this methodology, in each index ranking each algorithm receives a number of points, based on its location. These points are summed, and the final rankings are constructed in downward order.

Table 6: Example of order after score ranking

```
[('DBS', 42), ('SL', 36), ('PSC', 36), ('KM', 26)]
```

On the other hand, the method of winner ranking is based on the number of victories in a pairwise competition between the algorithms, taking all internal indices into account. The algorithm with the largest number of wins occupies the first place, and the algorithm with the smallest number of wins holds the last position. The number of victories can be calculated as follows:

$$\boldsymbol{V_i} = \sum_{j=1}^{m} |r| - \boldsymbol{p_{ij}}$$

Where $\boldsymbol{V_i}$ is the number of the victories of the *i*-th algorithm, *m* is the number of the internal indices, $|r|$ is the number of the algorithms in the ranking, and finally, $\boldsymbol{p_{ij}}$ is the rank position for the *i*-th algorithm in the *j*-th internal index.

Table 7: Example of order after winner ranking

```
[('DBS', 16), ('SL', 13), ('PSC', 13), ('KM', 8)]
```

Finally, the winner algorithm is found by calculating and finding the mean value of its position for all algorithms. Then it is stored and printed in the application as below:

Table 8: Example of stored table with results of clustering indices and the rank of each algorithm

	KM	SL	PSC	DBS
DB	0.816542	0.594282	0.612971	0.447282
HL	0.117292	0.158000	0.128158	0.131023
DU	0.021213	0.327290	0.039598	0.375184
CH	826.035171	1430.926201	1674.167609	1650.839885
SIL	0.482723	0.624418	0.562663	0.612571
Final Rank	4.000000	2.000000	3.000000	1.000000

Last but not least, we create a database instance locally, were we store the nine meta-attributes and the winner algorithm. The above procedure is followed for all 50 datasets (Appedix A), so at the end we have 50 entries stored in this local database which we will use later for our prediction tests.

	id	ma1	ma2	ma3	ma4	ma5	ma6	ma7	ma8	ma9	walgo
1	1	13,87267488	2	0	0,04	0	0	0,3662559401	0,0699893731	-0,07376050264	KM
2	2	13,92369888	4,392317423	28,57142857	0,2059202059	0	0,6191137083	0,1836213305	0,5962938683	2,802660971	PSC
3	3	13,87267488	3,584962501	0	0,1466666667	0	0	0,2846712951	-0,07185010282	-0,2768159307	KM
4	4	10,81378119	3,321928095	0	0,7222222222	0	0	0,2996128205	0,1266341513	-0,5855717996	KM
5	5	10,6617781	3,321928095	0	0,8641975309	0	0	0,3337549562	0,2538745844	-0,2481253325	DBS
6	6	10,71424552	3,321928095	0	0,8333333333	0	0	0,3213482935	0,2161907071	-0,2949133184	DBS
7	7	10,52160044	3,321928095	0	0,8843537415	0	0	0,3918961179	0,1273781423	-0,7534385712	KM
8	8	10,22881869	3,321928095	0	0,9166666667	0	0	0,3197225462	0,3852634348	-0,8195019301	DBS

Figure 11: 9 Meta-attributes and winner algorithm stored to local DB

4.3 DISTANCE MEASURES APPROACH

The *distance measures* approach assumes that data similarity can be noticed by extracting nineteen meta-attributes based on the object's attributes of a dataset. To better understanding how this works, we can consider the following definitions. Let assume that an Ω exists where $\Omega = \{1,\ldots,k,\ldots,n\}$ be a set of n instances indexed by k. Each instance k is represented by a vector of quantitative attributes $x_k = (x_{k,1},\ldots, x_{k,j},\ldots, x_{k,p})$ described by p attributes indexed by j where $x_{k,j} \in \Re$.

The distance measures approach has a time complexity $O(n^2)$, as it calculates for each row its distance to all the other rows. On the contrary in object's similarity approach this complexity is simply O(n) as it only computes statistical values like the percentage of the outliers. This difference in complexity, it has a great impact in performance for Big Data problems as it could be easily understand that for big

datasets size of 3 million rows, the operations would be around to $3.000.000^2$ for this $O(n^2)$ algorithm, which is actually a lot.

Firstly, the Euclidean distance between the instances in a dataset must be determined. The distance measured between instances x_k and x_l is given by the equation below:

$$d(x_k, x_l) = d_{k,l} = \sqrt{\sum_{j=1}^{p} (x_{k,l} - x_{l,j})^2}$$

Based on this measure, a vector **d**, containing the dissimilarity among all instances, is built as follows:

$$\mathbf{d} = [d_{1,2}, d_{1,3}, \ldots, d_{k,l}, \ldots, d_{n-2,\,n-1}, d_{n-1,\,n}]$$

Next, in the interval [0, 1], vector **d** is normalized, creating a new vector **w'**. Given a value indexed by *u* (**w**[*u*]) from vector **w**, the corresponding normalized value for index *u* (**w'**[*u*]) in vector **w'** is given by:

$$\mathbf{w'}[u] = \frac{w[u] - \min(w)}{\max(w) - \min(w)}$$

The 19 meta-features are extracted from each dataset after obtaining the vector **w'**. These meta-features (MD1 to MD19) are described in table 4.

According to A. Kalousis (2002) [44], histograms could provide more details about the characterizing data. In particular, from vector **w'**, MD1 to MD5 extract pure statistical data (measures, variance, standard deviation, skewness, and kurtosis). Meta-functions MD6 to MD15 capture vector **w'** dependent histogram information. This histogram occurs in the interval [0, 1], as the values from the vector **w'** are normalized. The meta-functions from MD16 to MD19 are extracted from the absolute *Z-score* histogram.

Table 9: Distance-based meta-features and their respective description.

Meta-attributes	***Description***
MD_1	Mean of **w'**
MD_2	Variance of **w'**
MD_3	Standard deviation of **w'**
MD_4	Skewness of **w'**
MD_5	Kurtosis of **w'**
MD_6	% of values in the interval [0,0.1]
MD_7	% of values in the interval (0.1,0.2]
MD_8	% of values in the interval (0.2,0.3]
MD_9	% of values in the interval (0.3,0.4]
MD_{10}	% of values in the interval (0.4,0.5]
MD_{11}	% of values in the interval (0.5,0.6]
MD_{12}	% of values in the interval (0.6,0.7]
MD_{13}	% of values in the interval (0.7,0.8]
MD_{14}	% of values in the interval (0.8,0.9]
MD_{15}	% of values in the interval (0.9,1.0]
MD_{16}	% of values with absolute Z-score in the interval [0,1)
MD_{17}	% of values with absolute Z-score in the interval [1,2)
MD_{18}	% of values with absolute Z-score in the interval [2,3)
MD_{19}	% of values with absolute Z-score in the interval [3,∞)

The Z-score [45] shows how many standard deviations an item is from the mean value of the distribution, and is measured as:

$$z = \frac{x - \mu}{\sigma}$$

Where x is the element value, μ is its mean value, and σ is its standard deviation. The absolute Z-score value is discretized into four bins, between the intervals [0,1), [1,2), [2,3) and [3,∞).

4.3.1 Extracting meta-features

One more as at the previous method, each dataset is imported to our system, and then all the data cleaning rules (mentioned on 4.1) are applied. As a second step, the dataset is normalized to the interval [0,1], and then we create an array that contains the Euclidean distance values between each row.

From this array, we can easily calculate the mean, the variation, the standard deviation, the skewness, and the kurtosis values and store these values to registers (MD_1 to MD_5). The next step is to calculate the histogram of this array with Euclidean distances and calculate / store requested meta-attributes MD_6 to MD_{15}. For the four last meta-attributes, we use the scipy library in order to calculate the Z-score first, then we calculate its absolute values and store them to registers MD_{16}-MD_{19}. The framework continues, with the creation of two tables where they will be stored the score of each clustering algorithm according to the clustering indices and the ranking between all the clustering algorithms.

Working identical to the fist method (4.2.1), after calculating and storing all the necessary meta-attributes, we proceed with running the clustering algorithms and clustering indices. For each dataset, we will execute the four clustering algorithms (DBSCAN, SL, PSO, K-Means), and they will be evaluated by the five clustering indices (DB, CH, SIL, DU, HL). The first two clustering algorithms (DBSCAN and SL) require as a configuration only the threshold. We compare the results for the number of clusters for each of them, and we are using this number of clusters as an input to the next algorithms (PSO, K-Means), which request this number as an input.

Identically to 4.2.1, we start by configuring and running the DBSCAN algorithm;

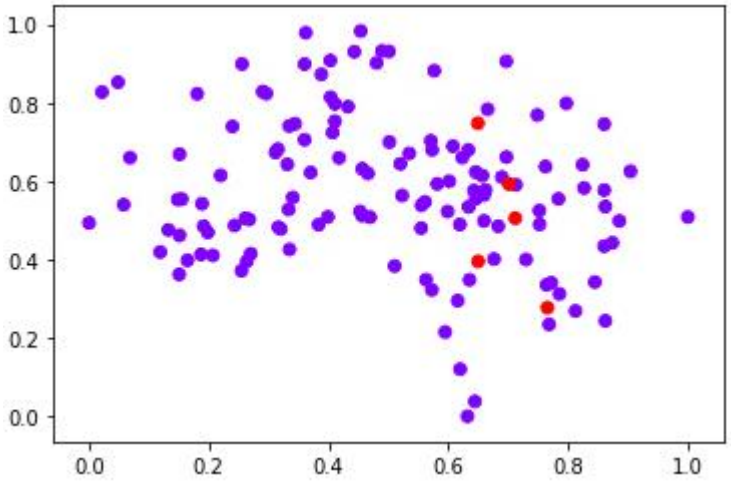

Figure 12: 2nd plot example for DBSCAN

After configuring the threshold (Appendix B), in above example DBSCAN suggests that the adequate number of clusters for this dataset is number 2. Then, we continue with the Single-Linkage algorithm. Also, at this specific dataset example, Single linkage suggests 4 clusters (Figure 13). We store and compare their scores according to the clustering indices (table 3), and we conclude to the ideal number of clusters, which in our example case was 2 number of clusters.

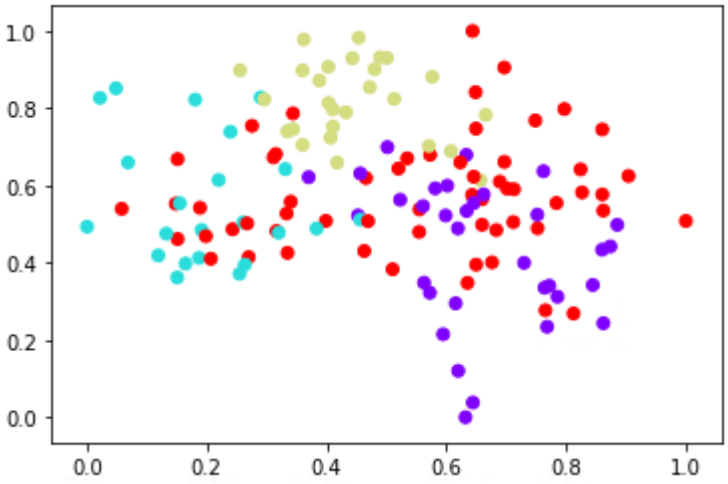

Figure 13: 2nd plot example for Single-Linkage

Later, this number of clusters is used as an input to the K-Means and PSO algorithm to store also their scores.

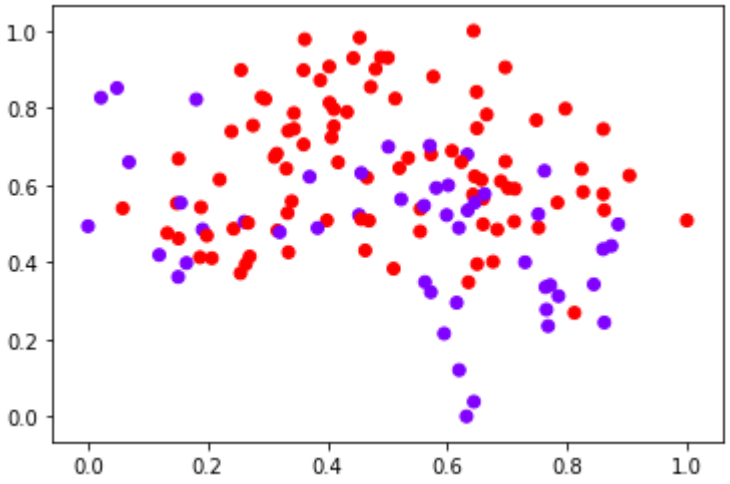

Figure 14: 2nd plot example for K-Means

We are completing the execution of all four clustering algorithms for each dataset we have stored their clustering indices values.

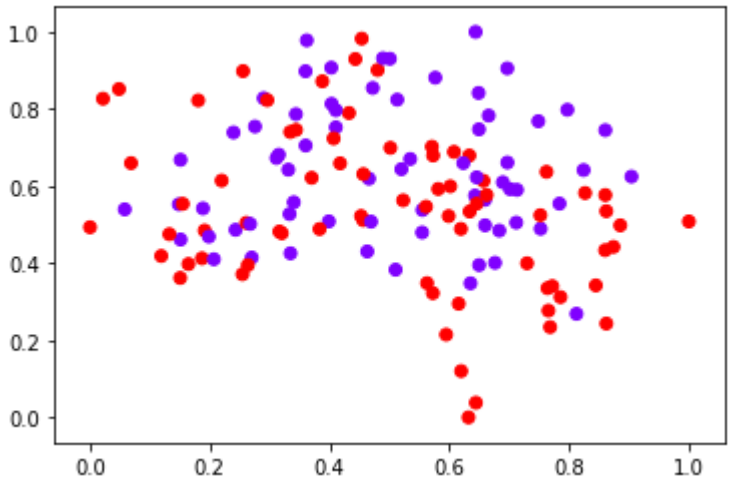

Figure 15: 2nd plot example for PSO

As we can see at the below example, all stored values of clustering indices can be easily compared in order to produce the socre and winner rank.

Table 10: 2nd example of stored clustering indices for all algorithms.

	KM	SL	PSC	DBS
DB	1.805096	1.795928	1.930328	1.211660
HL	0.000000	0.000000	0.000000	0.000000
DU	0.141194	0.218351	0.184676	0.316817
CH	34.874839	24.634592	35.237960	7.240550
SIL	0.190911	0.165070	0.185344	0.170980
Final Rank	0.000000	0.000000	0.000000	0.000000

There, again, we should consider one of them as a winning clustering algorithm according to the combination of score and winner ranking as below.

Table 11: 2nd example fo point system for score and winner ranking

```
{'CH': {'DBS': 4, 'KM': 8, 'PSC': 10, 'SL': 6},
 'DB': {'DBS': 10, 'KM': 6, 'PSC': 4, 'SL': 8},
 'DU': {'DBS': 10, 'KM': 4, 'PSC': 6, 'SL': 8},
 'HL': {'DBS': 4, 'KM': 10, 'PSC': 6, 'SL': 8},
 'SIL': {'DBS': 6, 'KM': 10, 'PSC': 8, 'SL': 4}}
```

To calculate the score ranking, we sum the scores of each algorithm, and we put it in descending order. So in our example, K-Means achieves a 38 score.

Table 12: 2nd example of order after score ranking

```
[('KM', 38), ('SL', 34), ('PSC', 34), ('DBS', 34)]
```

Following now the calculations to produce the winner ranking, according to point system, K-Means acquired 14 points.

Table 13: 2nd Example of order after winner ranking

```
[('KM', 14), ('SL', 12), ('PSC', 12), ('DBS', 12)]
```

We again calculate the mean of score and winner ranking position and store it. In both ranks, winner and score, K-Means managed to be the leader so it is obvious that this algorithm will be also the winner algorithm for this example dataset.

Table 14: 2nd example of stored table with results of clustering indices and the final rank of each algorithm

	KM	SL	PSC	DBS
DB	1.805096	1.795928	1.930328	1.211660
HL	0.000000	0.000000	0.000000	0.000000
DU	0.141194	0.218351	0.184676	0.316817
CH	34.874839	24.634592	35.237960	7.240550
SIL	0.190911	0.165070	0.185344	0.170980
Final Rank	1.000000	2.000000	3.000000	4.000000

Finally, we create a second database instance locally, where we store the nineteen meta-attributes and the winner algorithm. The above procedure is followed for all 50 datasets, so in the end, we have 50 entries stored in this local database, which we will use later for our prediction tests.

	md9	md10	md11	md12	md13	md14	md15	md16	md17	md18	md19	walgo
1	25,39268427	20,80110963	13,70033609	5,591350582	1,341186094	0,1412074331	0,004267804748	65,74591624	30,63805815	3,528791678	0,08723392905	KM
2	6,8942691	16,20341587	30,04352119	30,71243097	11,1911641	1,773762937	0,09435687379	71,01927367	24,28592327	3,833888015	0,8609150422	PSC
3	18,56947958	27,84333066	25,67045637	14,36694956	4,117726181	0,51561249	0,02638911129	67,31708567	28,35202562	4,19868695	0,1322017614	KM
4	15,05896958	11,65735568	8,131595282	14,60583489	22,26567349	12,29671012	1,148355059	59,41651148	39,39788951	1,185599007	0	KM
5	15,46660532	11,62487539	21,60110421	18,51085039	8,358254735	1,610305958	0,08434935971	63,10098919	35,34238172	1,556629093	0	DBS
6	13,70829769	11,62674651	10,11548332	15,67579127	17,67892786	10,68577131	2,088679783	61,51268891	37,5463359	0,9409751925	0	DBS
7	16,05628553	11,84418973	5,004193458	5,954710651	14,43481502	28,14276395	3,513186096	56,7794241	42,85714286	0,3634330445	0	KM
8	13,16526611	13,38935574	15,16806723	11,93277311	16,82072829	12,73109244	2,507002801	59,66386555	38,78151261	1,554621849	0	DBS

Figure 16: 19 Meta-attributes and winner algorithm stored to local DB

Chapter 5: Implementation Details

5.1 LANGUAGE OF CHOICE AND REASONING

As the problem of recommending a clustering algorithm belongs to the Machine Learning category, the use of *Python* as a programming language was the first choice. Python can be described as a programming language with a *lower barrier to entry*, *flexible*, has a *cross-platform compatibility*, and *good visualization options*.

The main reason, however, for this choice was that Python offers a vast library ecosystem. A library is a module or collection of modules that are released by different sources, such as PyPi, including a pre-written piece of code that enables users to access certain functionality or perform various actions. Python libraries have base-level objects, so developers don't always have to code them from the very start.

Machine Learning requires ongoing data processing, and the Python libraries allow you to access, manage, and transform data. Below are some of the libraries that were used in the framework:

- *Scikit-learn* for handling the Machine Learning algorithms of clustering.
- *Pandas* in order to create high-level data structures and analysis. Pandas library was responsible for importing data for their original format (*.txt or *.csv), data configuration, and exporting the results into a SQL schema.
- *Matplotlib* assisted in plotting the data into 2D plots, and as a result, this visualization helped in understanding the clusters better.
- *SciPy* offered some of the internal clustering indices as well as some of the clustering algorithms as well
- And last but not least, the *NumPy* library. Numpy handed the mathematical functions for the arrays, and it assisted in extracting and creating all the meta-attributes that were necessary for the implementation.

5.2 OPERATING SYSTEM, HARDWARE SPECIFICATIONS AND THEIR IMPACT

The entire framework was implemented in a Windows 10 environment as we took advantage of python's cross-platform compatibility. There, after installing the scientific Python distribution *Anaconda,* which is bundled in Spyder IDE, we could use Anacoda's *Jupyter* utility.

Jupyter Notebook is an Open-Source software that offered the opportunity to write "live" code with simultaneously executing and debugging every line of code. Also, it provided "live" charts and results.

The system on which the application was benchmarked was a 64-bit system, which may be needed when calculating the rational numbers to allow for larger integers and longs. On both cores, the processor is dual-core with a core frequency of 2GHz with 8 GB RAM.

5.3 CODE STRUCTURE

Following the typical approach, all functions were stored on a different notebook and were called at the declaration section at the main file. The code was arranged in such a way that the developer only needs to execute the main file line by line in order to observe the results of each command.

At the end of the execution file, a command responsible for inserting the extracted data into a SQL database was applied. Then after creating these three databases, one for each implementation, the final step was to execute the K-NN file that contains the K-NN algorithm and produces all the relevant results.

Each python command has notes and commends easy to read and understand. It is worth noticing that the entire process of adding extra algorithms (if needed) it can be easily performed. The developer can extend the experiment by simply calling any other cluster algorithm or clustering index he wants.

The most important parts of the implementation code can be found on Appendix D

Chapter 6: Results

After evaluating each clustering algorithm for all the existing datasets and storing their meta-attributes to a local database, what comes next is the evaluating of these two methods. On the one hand, there are the meta-attributes based on the dataset's attributes, and on the other hand, there are the meta-attributes based on the distance between the dataset's objects.

To evaluate each method, we treated this problem as a classification problem, and we performed the K-NN classification algorithm for the two datasets. For each of these datasets, we trained the system according to 75 percent of there total values. Then, with the remaining 25 percent, we tried to predict the winner algorithm for these values. Even if the datasets contained only 50 rows, the results were extremely positive.

We tried first to run the K-NN algorithm for the dataset that contained the meta-attributes based on the distance between its objects. Having the 75/25 ratio between the training and the test data, we managed to have a 77% accuracy in predicting the winner algorithm as it is depicted in the table below:

Table 15: Prediction accuracy for the distance-based method

	precision	recall	f1-score	support
DBS	1.00	0.67	0.80	3
KM	0.80	0.80	0.80	5
PSC	1.00	0.50	0.67	2
SL	0.60	1.00	0.75	3
accuracy			0.77	13
macro avg	0.85	0.74	0.75	13
weighted avg	0.83	0.77	0.77	13

The results reveal that there is an excellent correlation between the distance of each object and the clustering algorithm that provides the most accurate clusters.

On the other side, when the K-NN algorithm was executed for the dataset that included the meta-attributes based on the attributes, with the analogy between training data and test data to be 75/25, we had only 46% accuracy in predicting the winner algorithm.

Table 16: Prediction accuracy for the attributes-based method (1st attempt)

	precision	recall	f1-score	support
DBS	0.50	0.33	0.40	3
KM	0.33	0.50	0.40	4
PSC	1.00	0.67	0.80	3
SL	0.33	0.33	0.33	3
accuracy			0.46	13
macro avg	0.54	0.46	0.48	13
weighted avg	0.53	0.46	0.48	13

Trying to improve this rating, we increased the number of training data to 80%. Then with the remaining 20%, we tried the next prediction.

Table 17: Prediction accuracy for the attributes-based method (2nd attempt)

	precision	recall	f1-score	support
DBS	0.50	1.00	0.67	1
KM	0.50	1.00	0.67	2
PSC	1.00	0.75	0.86	4
SL	1.00	0.33	0.50	3
accuracy			0.70	10
macro avg	0.75	0.77	0.67	10
weighted avg	0.85	0.70	0.69	10

The accuracy of this prediction increased significantly to 70%. The above shows us that this method that contains the nine meta-attributes requests a dataset that includes more values from the existing one, which will adequately train the system, and as a result, the prediction later will become more accurate.

Comparing those two methods, we realize that both approaches are efficient with high recommendation quality. Both results showed the feasibility of meta-learning systems for an unlabeled approach to the selection problem of the clustering algorithms.

Characterization based on meta-attributes that do not depend on the object labels and evaluation of the algorithm by using internal indices that avoid known solutions allow the methodology proposed to be applied to any clustering problem. However, as the results reveal, the meta-attributes based on the distance between the objects are more efficient than the meta-attributes based on datasets attributes. With a slight

difference (77% from 70%), the second method produced better results in predicting the most suitable cluster algorithm.

Lastly, it is worth to be mentioned, that for Big Data problems, the first approach has a lot more value on that as it overcomes the algorithm complexity of the distance-based approach.

Chapter 7: Conclusions and Future Work

This Master Thesis investigated the automatic recommendation of Clustering Algorithms. Data characterization is an essential part of meta-learning-based recommender systems. To improve the proposal, two methods were experimentally examined in a set of 50 datasets.

The suggested collection of meta-functions incorporates the tests of correlation and dissimilarity. Using this new set of meta-functions, experiments were performed to test the predictive efficiency of a clustering algorithm recommender program. The framework recommends the most acceptable ranking of clustering algorithms for a new dataset.

The assessment occurred on two occasions. The first was by obtaining some meta-attributes according to dataset's statistic values, and the second was by obtaining these meta-attributes by the distanced base model. After extracting their meta-data, each dataset examined in order to find the most suitable clustering algorithm according to metrics from clustering internal indices. These two pieces of information, meta-data, and the clustering algorithm, were stored in local databases and then tries for prediction occurred.

According to the K-NN algorithm, both methods were extraordinarily efficient and accurate. However, a comparison between them depicts that the method based on the distance between the dataset's objects was more accurate, with also a smaller training set of data. It presented better results than the classic approach along with a higher recommendation quality. Finally, a selection technique of the attributes could be used for the meta-features collected by these two methods.

As future work, new meta-features could be introduced and experimentally tested, removing other elements from the datasets. Using more datasets could make the meta-learners more accurate. Also, the recommender method might also use different classification algorithms.

It is interesting to consider how the metalearing approach could be adapted or extended also to problems in other domains, such as operations research and of course how metalearning could be possible have an entity in *Big Data* problems.

Bibliography

[1] Gan, G., Ma, C., Wu, J., 2007. Data clustering: theory, algorithms, and applications. SIAM.

[2] J.R. Rice, The algorithm selection problem, Adv. Comp. 15 (1976) 65–118.

[3] K.A. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv. 41 (1) (2009) 1–25, http://dx.doi.org/10.1145/1456650.1456656.

[4] K.A. Smith-Miles, Towards insightful algorithm selection for optimisation using meta-learning concepts, in: IEEE International Joint Conference on Neural Networks, 2008, IJCNN 2008 (IEEE World Congress on Computational Intelligence), 2008, pp. 4118–4124.

[5] B. Arinze, S.-L. Kim, M. Anandarajan, Combining and selecting forecasting models using rule based induction, Comput. Oper. Res. 24 (1997) 423–433.

[6] H. Guo, Algorithm Selection for Sorting and Probabilistic Inference: A Machine Learning-Based Approach, Kansas State University, 2003 (256).

[7] Daniel Gomes Ferrari, Leandro Nunes de Castro, Clustering Algorithm Selection By Meta-Learning Systems: A new distance-based problem characterization and ranking combination methods, Information Sciences [2015]

[8] Wikipedia, K-Nearest neigbors Algorithm

[9] C. M. Bishop, “Pattern recognition,” Machine Learning, vol. 128, 2006.

[10] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013

[11] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, 2015

[12] M. T. Banday and T. R. Jan, “Effectiveness and limitations of statistical spam filters,” arXiv, 2009.

[13] P. Stone and M. Veloso, “Multiagent systems: A survey from a machine learning perspective,” Autonomous Robots, vol. 8, no. 3, pp. 345–383, 2000

[14] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” 2007

[15] T. Hastie, R. Tibshirani, and J. Friedman, “Unsupervised learning,” in The elements of statistical learning. Springer, 2009, pp. 485–585

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” Journal of artificial intelligence research, vol. 4, pp. 237–285, 1996

[17] G. A. Seber and A. J. Lee, Linear regression analysis. John Wiley & Sons, 2012, vol. 936

[18] Zubin, Joseph (1938). "A technique for measuring like-mindedness". The Journal of Abnormal and Social Psychology. 33 (4): 508–516. doi:10.1037/h0055441. ISSN 0096-851X.

[19] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. CiteSeerX 10.1.1.121.9220.

[20] Everitt B (2011). Cluster analysis. Chichester, West Sussex, U.K: Wiley. ISBN 9780470749913.

[21] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07

[22] Principles of Geographical Information Systems, By Peter A. Burrough, Rachael McDonnell, Rachael A. McDonnell, Christopher D. Lloyd

[23] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: An Introduction to Information Retrieval, page 163-167. Cambridge University Press, 2009.

[24] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[25] Kennedy, J.; Eberhart, R.C. (2001). Swarm Intelligence. Morgan Kaufmann.

[26] Bonyadi, M. R.; Michalewicz, Z. (2017). "Particle swarm optimization for single objective continuous space problems: a review". Evolutionary Computation. 25 (1): 1–54 doi:10.1162/EVCO_r_00180

[27] Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2002. Clustering Validity Methods Part 1. ACM SIGMOD Record 31(2).

[28] Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2002. Clustering Validity Methods Part 2. ACM SIGMOD Record 31(3).

[29] Rendón, E.; Abundez, I.; Arizmendi, A.; Quiroz, E.M. Internal versus external cluster validation indexes. Int. J. Comput. Commun. 2011, 5, 27–34.

[30] Kuncheva, L.I.; Vetrov, D.P. Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Trans. Pattern Anal. Mach. Intell. 2006, 28, 1798–1808

[31] Davies, David L.; Bouldin, Donald W. (1979). "A Cluster Separation Measure". IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2): 224–227. doi:10.1109/TPAMI.1979.4766909.

[32] Hubert, L.J., & Levin, J.R. (1976). A general statistical framework for assessing categorical clustering in free recall.Phycological Bulletin, 83, 1072–1080.

[33] Dunn, J. C. (1973-09-17). "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". Journal of Cybernetics. 3 (3): 32–57. doi:10.1080/01969727308546046

[34] Calinski, T., and Harabasz, J. (1974) A Dendrite Method for Cluster Analysis, Communications in Statistics, 3, 1-27.

[35] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics. 20: 53–65. doi:10.1016/0377-0427(87)90125-7.

[36] J. R. Rice, “The algorithm selection problem" Advances in Computers, vol. 15, pp. 65-118, 1976.

[37] B. A. Huberman, R.M. Lukose and T. Hogg, “An economic approach to hard computational problems” Science, vol. 275, pp. 51-54, 1996.

[38] L. Xu, F. Hutter, H. H. Hoos and K. Leyton-Brown, “SATzilla-07: The Design and Analysis of an Algorithm Portfolio for SAT” in Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP-07), pp. 712-727, 2007.

[39] Carla P. Gomes, Bart Selman, Nuno Crato and Henry Kautz, “Heavy-tailed phenomena in satisfiability and constraint satisfaction problems” J. Autom. Reason., vol. 24(1-2), pp. 67-100, 2000.

[40] C. P. Gomes and B. Selman, “Algorithm portfolios” Artificial Intelligence, vol. 126(1-2), pp. 43-62, 2001.

[41] Ronald Deep, “Probability and Statistics: With Integrated Software Routines”, Academic Press, 2006

[42] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2004) 85–126.

[43] "SPSS Tutorials: Pearson Correlation". Retrieved 14 May 2017

[44] Kalousis, A., 2002. Algorithm selection via meta-learning. University of Geneva, Genebra.

[45] Z-score definition, Adam Hayes, https://www.investopedia.com/terms/z/zscore.asp

Appendices

Appendix A: List of Datasets used

Dataset	rows	attributes	DBSCAN (eps/min)	SL
3d_spatial_network	3750	4	0.11, 15	ward, 10
absenteeism_at_work	740	21	1.1, 4	compl, 2.1
ae_train	1250	12	0.3, 4	ward, 11
c1r4r_02	180	10	0.44, 4	ward, 4
c1r5r_01	162	10	0.49, 4	ward, 4
c1r5r_02	168	10	0.46, 4	ward, 4
c1r6r_02	147	10	0.4, 4	ward, 4
c1r7r_02	120	10	0.5, 4	ward, 3.5
Frogs_MFCCs	700	12	0.35, 4	ward, 7
gesture_phase_b3_raw	750	12	0.32, 4	compl, 1.5
HTRU_2	1700	9	0.41, 4	ward, 5
l1n_01	136	10	0.33, 4	average, 0.9
l1n_05	131	10	0.4, 4	average, 0.9
l1r_02	134	10	0.4, 5	average, 0.9
l1r_03	135	10	0.4, 5	average, 0.9
l1r_04	138	10	0.37, 5	average, 0.9
l1r_05	124	10	0.38, 4	average, 0.9
LG_G-Watch_1	2100	6	0.107, 6	average, 0.9
movement_libras_1	45	91	2, 4	ward, 5
movement_libras_5	90	91	2.5, 4	ward, 10
movement_libras_10	165	12	0.45, 4	ward, 3
mturk_cluster_data	85	180	3, 3	ward, 13
mturk_data_feature	30	500	7, 7	average, 8.9
perfume_dataset	20	29	0.3, 2	ward, 2
Samsung-Galaxy-Gear-2	2100	6	0.107, 6	average, 0.9
SCADI	70	143	5, 6	ward, 12
seeds_dataset	210	8	0.5, 4	ward, 2.5
turkiye-student-evaluation	455	33	1, 4	ward, 15
Wholesale_customers_data	440	8	1, 4	ward, 7
zAggregation	788	3	0.1, 5	ward, 3.5
zCollege	777	17	0.3, 4	ward, 6
zD	3100	3	0.1, 4	ward, 9
zflu	240	3	0.1, 4	ward, 3
zGENERAL	8950	17	0.3, 4	ward, 30
zMall_Customers	200	4	0.15, 4	ward, 3
zs3	5000	2	0.02, 4	ward, 7
ztoys	373	3	0.2, 4	single, 0.5
zunb	6500	2	0.07, 4	ward, 8

zcars	261	7	0.23, 4	ward, 5
buddymove_holidayiq	249	7	0.27, 4	ward, 4
c1r4r_01	175	10	0.4, 4	ward, 5
c1r6r_01	146	10	0.5, 4	ward, 3.5
c1r7r_01	117	10	0.5, 4	ward, 3.5
gesture_phase_a2_raw	750	12	0.32, 4	ward, 6
gesture_phase_c3_raw	750	12	0.33, 4	ward, 6
l1n_02	138	10	0.38, 4	ward, 6
l1n_04	131	10	0.5, 4	ward, 3
l1r_01	139	10	0.38, 4	ward, 3
movement_libras_8	135	91	2, 4	ward, 4
Sales_Transactions_Dataset	141	12	0.6, 4	ward, 3

Appendix B: Extracted meta-features at object's similarity approach

id	ma1	ma2	ma3	ma4	ma5	ma6	ma7	ma8	ma9	walgo
1	13.873	2.000	0.000	0.040	0.000	0.000	0.366	0.070	-0.074	KM
2	13.924	4.392	28.571	0.206	0.000	0.619	0.184	0.596	2.803	PSC
3	13.873	3.585	0.000	0.147	0.000	0.000	0.285	-0.072	-0.277	KM
4	10.814	3.322	0.000	0.722	0.000	0.000	0.300	0.127	-0.586	KM
5	10.662	3.322	0.000	0.864	0.000	0.000	0.334	0.254	-0.248	DBS
6	10.714	3.322	0.000	0.833	0.000	0.000	0.321	0.216	-0.295	DBS
7	10.522	3.322	0.000	0.884	0.000	0.000	0.392	0.127	-0.753	KM
8	10.229	3.322	0.000	0.917	0.000	0.000	0.320	0.385	-0.820	DBS
9	13.036	3.585	0.000	0.250	0.000	0.000	0.281	-0.089	-0.240	KM
10	13.136	3.585	0.000	0.233	0.000	0.000	0.287	-0.084	-0.231	PSC
11	13.901	3.170	11.111	0.118	0.000	0.858	0.485	2.240	12.571	PSC
12	10.409	3.322	0.000	1.176	0.000	0.000	0.247	0.094	19.338	SL
13	10.355	3.322	0.000	1.221	0.000	0.000	0.269	0.782	15.244	SL
14	10.388	3.322	0.000	1.269	0.000	0.000	0.306	-0.061	9.468	DBS
15	10.399	3.322	0.000	1.259	0.000	0.000	0.268	-0.368	14.567	DBS
16	10.430	3.322	0.000	1.232	0.000	0.000	0.303	-0.394	8.663	SL
17	10.276	3.322	0.000	1.371	0.000	0.000	0.308	0.162	9.207	SL
18	13.621	2.585	0.000	0.071	0.000	0.000	0.392	0.026	-0.581	KM
19	12.000	6.508	0.000	2.637	0.000	0.000	0.333	-0.221	-0.428	DBS
20	13.000	6.508	0.000	1.319	0.000	0.000	0.322	-0.458	-0.589	SL
21	10.951	3.585	0.000	0.909	0.000	0.000	0.296	-0.048	-0.371	SL
22	13.901	7.492	100.000	2.281	0.000	0.784	0.000	0.000	0.000	PSC
23	13.873	8.966	100.000	5.980	0.000	0.388	0.000	0.000	0.000	PSC
24	9.180	4.858	0.000	9.828	0.000	0.000	0.941	-0.272	1.174	SL
25	13.621	2.585	0.000	0.071	0.000	0.000	0.641	-0.112	-0.692	KM
26	13.289	7.160	99.301	2.428	0.000	0.721	1.000	0.379	-0.452	PSC
27	10.714	3.000	12.500	0.595	0.000	0.333	0.667	0.267	-0.732	PSC
28	13.874	5.044	33.333	0.233	0.000	0.292	0.682	-0.198	-1.000	KM
29	11.781	3.000	25.000	0.398	0.000	0.560	0.416	5.149	50.252	SL
30	11.207	1.585	33.333	0.127	0.000	0.218	0.508	0.227	-1.358	DBS
31	13.689	4.087	0.000	0.250	0.000	0.000	0.343	1.709	10.330	KM
32	13.183	1.585	0.000	0.032	0.000	0.000	0.426	-0.030	-1.220	DBS
33	9.492	1.585	33.333	0.417	0.000	0.538	0.508	0.067	-0.750	PSC
34	17.215	4.087	58.824	0.021	0.000	0.342	0.389	5.210	71.740	DBS
35	9.644	2.000	0.000	0.625	0.000	0.000	0.421	0.190	-0.699	KM
36	13.288	1.000	0.000	0.020	0.000	0.000	0.669	0.132	-1.075	PSC
37	10.128	1.585	33.333	0.268	0.000	0.615	0.721	0.230	-0.756	PSC
38	13.666	1.000	0.000	0.031	0.000	0.000	0.568	1.265	4.727	PSC
39	10.835	2.807	14.286	0.438	0.000	0.359	0.677	0.487	-0.387	KM
40	10.767	2.807	0.000	0.574	0.000	0.000	0.594	0.485	-0.376	KM
41	10.773	3.322	0.000	0.800	0.000	0.000	0.284	0.352	0.051	SL
42	10.512	3.322	0.000	0.959	0.000	0.000	0.336	0.167	-0.491	DBS
43	10.192	3.322	0.000	1.111	0.000	0.000	0.333	0.265	-0.555	DBS
44	13.136	3.585	0.000	0.211	0.000	0.000	0.276	-0.076	-0.320	SL
45	13.136	3.585	0.000	0.233	0.000	0.000	0.288	-0.085	-0.322	KM
46	10.430	3.322	0.000	1.087	0.000	0.000	0.268	0.581	14.925	KM
47	10.355	3.322	0.000	1.221	0.000	0.000	0.258	0.761	15.146	SL
48	10.441	3.322	0.000	1.295	0.000	0.000	0.268	-0.535	14.729	KM
49	13.585	6.508	0.000	0.920	0.000	0.000	0.298	-0.395	-0.528	DBS
50	10.725	3.585	0.000	1.123	0.000	0.000	0.296	-0.071	-0.227	KM

Appendix C: Extracted meta-features at distance-based approach

id	md1	md2	md3	md4	md5	md6	md7	md8	md9	md10	md11	md12	md13	md14	md15	md16	md17	md18	md19	walgo
1	0.570	0.050	0.224	0.212	-0.399	1.912	10.508	20.607	25.393	20.801	13.700	5.591	1.341	0.141	0.004	65.746	30.638	3.529	0.087	KM
2	1.708	0.146	0.382	-0.613	0.790	0.207	0.806	2.074	6.894	16.203	30.044	30.712	11.191	1.774	0.094	71.019	24.286	3.834	0.861	PSC
3	0.815	0.045	0.212	-0.048	-0.209	0.130	1.594	7.166	18.569	27.843	25.670	14.367	4.118	0.516	0.026	67.317	28.352	4.199	0.132	KM
4	1.140	0.171	0.414	-0.272	-1.173	0.521	4.221	10.093	15.059	11.657	8.132	14.606	22.266	12.297	1.148	59.417	39.398	1.186	0.000	KM
5	1.129	0.170	0.412	-0.174	-0.951	1.196	7.737	13.810	15.467	11.625	21.601	18.511	8.358	1.610	0.084	63.101	35.342	1.557	0.000	DBS
6	1.139	0.187	0.433	-0.193	-1.094	1.219	6.401	10.800	13.708	11.627	10.115	15.676	17.679	10.686	2.089	61.513	37.546	0.941	0.000	DBS
7	1.166	0.227	0.477	-0.231	-1.459	0.308	3.439	11.304	16.056	11.844	5.004	5.955	14.435	28.143	3.513	56.779	42.857	0.363	0.000	KM
8	1.190	0.175	0.418	-0.174	-0.965	0.980	4.314	8.992	13.165	13.389	15.168	11.933	16.821	12.731	2.507	59.664	38.782	1.555	0.000	DBS
9	0.866	0.049	0.221	-0.094	-0.175	0.116	1.449	6.422	16.392	26.620	26.913	16.220	5.073	0.740	0.056	67.595	28.020	4.233	0.152	KM
10	0.851	0.050	0.224	0.035	-0.159	0.121	1.548	7.612	19.854	28.551	24.975	12.929	3.802	0.569	0.039	67.618	27.998	4.202	0.182	PSC
11	0.484	0.171	0.414	1.392	0.911	37.292	33.875	6.914	4.854	8.469	5.242	2.666	0.559	0.127	0.002	80.093	13.224	5.996	0.687	PSC
12	0.804	0.069	0.263	0.866	3.048	1.307	9.292	22.876	31.100	25.370	7.930	0.654	0.000	0.752	0.719	72.190	24.924	1.416	1.471	SL
13	0.821	0.080	0.284	0.590	1.629	2.079	10.711	20.435	28.597	24.674	10.100	1.856	0.164	0.881	0.505	70.851	26.142	1.515	1.491	SL
14	0.873	0.094	0.307	1.183	3.520	1.234	8.619	22.007	36.012	22.904	5.084	1.190	0.943	1.212	0.797	76.961	18.887	1.784	2.368	DBS
15	0.849	0.067	0.259	0.594	2.370	1.061	6.877	18.032	32.968	28.358	10.404	0.818	0.243	0.818	0.420	72.029	24.356	2.134	1.481	DBS
16	0.852	0.094	0.307	1.005	2.440	1.619	9.923	24.236	27.124	24.289	8.590	1.312	0.360	1.492	1.058	74.537	21.908	1.058	2.497	SL
17	0.868	0.104	0.322	1.261	3.372	1.665	12.720	28.101	34.566	14.175	5.180	0.970	1.298	0.682	0.643	74.889	20.968	1.941	2.203	SL
18	0.704	0.129	0.359	0.675	-0.304	2.988	19.830	24.796	18.280	13.231	9.574	6.512	3.791	0.988	0.010	66.699	28.793	4.478	0.029	KM
19	3.333	1.034	1.017	-0.155	-0.066	1.212	3.131	6.061	13.737	18.586	20.909	19.394	10.000	5.253	1.717	68.889	26.162	4.949	0.000	DBS
20	3.298	0.982	0.991	-0.445	0.066	1.174	3.171	5.293	7.166	16.579	23.895	20.949	12.809	7.765	1.199	69.588	25.418	4.894	0.100	SL
21	0.989	0.064	0.253	-0.104	-0.273	0.177	1.382	6.548	15.085	23.112	25.358	19.135	7.354	1.715	0.133	67.214	28.655	4.050	0.081	SL
22	5.996	3.307	1.818	0.454	0.217	0.028	1.148	4.034	18.992	20.924	27.535	15.014	5.770	4.398	2.157	70.336	24.286	5.322	0.056	PSC
23	8.672	0.734	0.857	0.472	0.051	2.069	5.287	13.563	24.828	21.609	14.713	6.897	7.126	2.529	1.379	69.425	25.747	4.598	0.230	PSC
24	1.452	0.976	0.988	1.137	1.271	15.263	27.895	21.053	13.158	9.474	5.789	5.263	0.526	0.526	1.053	73.158	22.105	3.158	1.579	SL
25	0.710	0.162	0.402	0.683	-0.348	6.230	22.352	20.907	15.839	12.231	9.351	6.567	4.337	1.982	0.205	66.055	29.459	4.464	0.021	KM
26	6.185	1.423	1.193	-1.144	0.636	0.373	0.621	1.615	3.520	6.004	9.027	8.240	13.458	32.878	24.265	72.671	21.781	4.720	0.828	PSC
27	0.988	0.209	0.458	-0.145	-0.997	5.140	13.579	11.032	8.827	11.251	22.005	15.147	8.904	3.377	0.738	61.294	37.781	0.925	0.000	PSC
28	2.429	1.109	1.053	0.812	0.397	0.793	5.737	22.347	23.245	19.197	12.086	6.873	5.587	1.241	2.893	71.625	23.919	4.396	0.060	KM
29	0.796	0.217	0.466	-0.258	-1.201	20.597	7.992	10.684	0.702	29.627	17.443	9.023	3.375	0.466	0.090	60.494	38.999	0.507	0.000	SL
30	0.637	0.091	0.301	-0.397	-0.947	7.387	9.908	7.500	7.527	12.616	16.476	17.863	15.132	5.102	0.489	61.626	37.740	0.634	0.000	DBS
31	0.822	0.098	0.313	0.785	0.451	1.877	18.058	30.097	24.679	13.541	6.792	3.541	1.204	0.192	0.019	69.627	25.632	4.074	0.667	KM
32	0.631	0.064	0.253	-0.020	-0.129	3.527	4.923	11.298	20.504	21.189	17.968	12.391	5.548	2.018	0.635	67.490	26.737	5.773	0.000	DBS
33	0.724	0.178	0.422	-0.152	-1.618	11.789	15.143	10.875	7.354	5.418	2.699	4.732	31.688	9.568	0.736	55.220	44.780	0.000	0.000	PSC
34	1.065	0.192	0.438	-0.094	-0.689	4.399	12.900	19.512	22.524	24.317	13.376	2.721	0.217	0.031	0.002	64.538	32.797	2.625	0.040	DBS
35	0.681	0.082	0.286	0.147	-0.280	4.261	10.658	15.397	22.271	22.271	14.538	6.814	2.834	0.864	0.090	67.201	28.920	3.784	0.095	KM
36	0.396	0.040	0.199	0.335	-0.389	7.599	17.125	21.949	20.356	16.986	9.994	4.337	1.395	0.249	0.011	65.569	31.427	2.859	0.145	PSC
37	0.662	0.207	0.455	0.266	-1.519	13.757	15.000	14.529	15.225	2.900	0.000	2.335	16.132	15.202	4.919	52.754	47.246	0.000	0.000	PSC
38	0.217	0.058	0.241	1.808	2.204	28.518	51.172	5.823	0.080	0.068	2.930	3.620	4.749	2.448	0.592	85.602	4.288	8.452	1.657	PSC
39	0.865	0.170	0.412	0.372	-0.619	4.889	13.879	20.973	19.994	13.212	12.841	9.231	3.855	1.014	0.112	63.882	33.369	2.703	0.047	KM
40	0.796	0.126	0.355	0.432	-0.503	2.044	11.643	21.253	19.718	16.948	12.165	9.389	5.085	1.561	0.194	65.073	31.578	3.300	0.049	KM
41	1.081	0.167	0.409	-0.333	-1.169	1.071	6.883	12.512	11.987	9.741	10.818	16.841	24.335	5.471	0.342	57.511	41.406	1.084	0.000	SL
42	1.138	0.186	0.431	-0.260	-1.172	0.529	4.695	12.083	15.239	12.508	10.345	18.526	22.617	3.174	0.283	60.274	38.479	1.247	0.000	DBS
43	1.121	0.183	0.428	-0.015	-1.135	0.324	3.581	10.669	17.006	15.178	13.454	10.772	17.846	10.492	0.678	57.928	41.424	0.648	0.000	DBS
44	0.884	0.050	0.224	-0.135	-0.172	0.161	1.572	6.202	15.775	26.059	26.440	17.349	5.523	0.872	0.048	67.610	28.005	4.240	0.146	SL
45	0.865	0.049	0.222	-0.083	-0.157	0.133	1.646	6.963	17.767	27.771	26.390	14.649	4.105	0.539	0.038	67.725	27.842	4.279	0.154	KM
46	0.907	0.086	0.293	0.504	0.742	0.582	5.988	16.249	24.162	26.764	17.709	5.162	1.269	1.576	0.540	70.168	25.643	3.142	1.047	KM
47	0.877	0.075	0.275	0.456	1.140	0.540	5.931	14.985	25.015	28.268	17.945	5.285	0.575	0.411	1.045	70.065	26.330	2.278	1.327	SL
48	0.843	0.071	0.266	0.809	3.017	1.522	9.342	22.406	32.239	25.107	7.371	0.542	0.219	0.657	0.594	72.391	24.502	1.668	1.439	KM
49	3.089	0.840	0.916	-0.251	-0.064	0.951	3.814	7.518	14.748	25.340	24.245	15.202	6.766	1.338	0.077	68.701	26.412	4.743	0.144	DBS
50	1.002	0.068	0.261	-0.071	-0.289	0.162	1.510	6.869	15.684	23.799	25.258	17.903	7.173	1.510	0.132	67.082	28.825	3.972	0.122	KM

Appendix D: System's important sections in python

Data Preparation

```
# Data import

dataP = pd.read_csv("datasets/training/3d_spatial_network.txt", sep = " ", header = None)


# Cleaning data from columns with same values or totally different
# and Discover the discrete attributes from the dataset
discrete = 0
entr = []
concetr = []
dataContinuous = dataP

for i in range(0,dataP.shape[1]):
    perc = dataP[i].value_counts(normalize=True)*100

    if perc.values[0] == 100:
        dataP = dataP.drop([i],axis = 1)
        dataContinuous = dataContinuous.drop([i],axis = 1)
        print('found same')
    elif perc.values[0] == (1/dataP.shape[0]):
        dataP = dataP.drop([i],axis = 1)
        dataContinuous = dataContinuous.drop([i],axis = 1)
        print('found totally different')

    if  30 <= perc.values[0] <100:
        print('Column ', i, 'has discrete objects')
        discrete += 1
        # calculate the entropy
        entr.append(pandas_entropy(i))
        # calculate the concetration
        concetr.append(concentration(dataP, i))
        # separates the dataset to Continuous and Discrete
        dataContinuous = dataContinuous.drop([i],axis = 1)
```

Meta Attributes (9)

```
total_obj = dataP.size
ma1 = math.log(total_obj, 2.0)

total_attr = len(dataP.columns)
ma2 = math.log(total_attr, 2.0)

# Calculating Outliers based on IQR method
Q1 = dataP.quantile(0.25)
Q3 = dataP.quantile(0.75)
IQR = Q3 - Q1
dataOutliers = (dataP < (Q1 - 1.5 * IQR)) |(dataP > (Q3 + 1.5 * IQR))
d = dataOutliers.nunique()

outliers = sum(d)

ma4 = (outliers / total_obj)*100

ma3 = (discrete / total_attr)*100

if discrete == 0:
    ma5 = 0
    ma6 = 0
else:
    ma5 = np.mean(entr)
    ma6 = np.mean(concetr)

correlation = np.absolute(dataContinuous.corr(method = 'pearson'))
if correlation.empty :
    correlation = 0
```

```
correl = np.mean(correlation)

ma7 = np.mean(correl)

skewness = dataContinuous.skew(axis = 0, skipna = True)
if skewness.empty :
    skewness = 0

ma8 = np.mean(skewness)

kurt = dataContinuous.kurtosis()
if kurt.empty :
    kurt = 0

ma9 = np.mean(kurt)

rankingsTable = pd.DataFrame.from_items([('DB',[0, 0, 0, 0, 0, 0, 0]),
                                         ('HL', [0, 0, 0, 0, 0, 0, 0]), ('DU', [0, 0, 0, 0, 0, 0
, 0]),
                                         ('CH', [0, 0, 0, 0, 0, 0, 0]), ('SIL', [0, 0, 0, 0, 0,
0, 0]),('Final Rank', [0, 0, 0, 0, 0, 0, 0]), ],
                                        orient = 'index', columns = ['KM', 'SL', 'PSC', 'DBS'])

scoresTable = pd.DataFrame.from_items([('DB',[0, 0, 0, 0, 0, 0, 0]),
                                         ('HL', [0, 0, 0, 0, 0, 0, 0]), ('DU', [0, 0, 0, 0, 0, 0
, 0]),
                                         ('CH', [0, 0, 0, 0, 0, 0, 0]), ('SIL', [0, 0, 0, 0, 0,
0, 0]),('Final Rank', [0, 0, 0, 0, 0, 0, 0])],
                                        orient = 'index', columns = ['KM', 'SL', 'PSC', 'DBS'])
```

Meta Attributes (19)

```
d = scipy.spatial.distance.pdist(dataNP, metric='euclidean')

d_len = len(d)

md1 = np.mean(d)
md2 = np.var(d)
md3 = np.std(d)
md4 = scipy.stats.skew(d)
md5 = scipy.stats.kurtosis(d)

d_hist = np.histogram(d)

x = d_hist[0]

md6 = (x[0]/sum(x))*100
md7 = (x[1]/sum(x))*100
md8 = (x[2]/sum(x))*100
md9 = (x[3]/sum(x))*100
md10 = (x[4]/sum(x))*100
md11 = (x[5]/sum(x))*100
md12 = (x[6]/sum(x))*100
md13 = (x[7]/sum(x))*100
md14 = (x[8]/sum(x))*100
md15 = (x[9]/sum(x))*100

#Z-Score = (d[0]-md1) / md3
z = scipy.stats.zscore(d)

y1 = sum(0 <= x < 1 for x in np.absolute(z))
y2 = sum(1 <= x < 2 for x in np.absolute(z))
y3 = sum(2 <= x < 3 for x in np.absolute(z))
y4 = sum(3 <= x for x in np.absolute(z))
y = y1 + y2 + y3 + y4

md16 = (y1/y)*100
md17 = (y2/y)*100
md18 = (y3/y)*100
md19 = (y4/y)*100

rankingsTable = pd.DataFrame.from_items([('DB',[0, 0, 0, 0, 0, 0, 0]),
                                         ('HL', [0, 0, 0, 0, 0, 0, 0]), ('DU', [0, 0, 0, 0, 0, 0
, 0]),
```

```
                                        ('CH', [0, 0, 0, 0, 0, 0, 0]), ('SIL', [0, 0, 0, 0, 0,
0, 0]),('Final Rank', [0, 0, 0, 0, 0, 0, 0]), ],
                                       orient = 'index', columns = ['KM', 'SL', 'PSC', 'DBS'])

scoresTable = pd.DataFrame.from_items([('DB',[0, 0, 0, 0, 0, 0, 0]),
                                        ('HL', [0, 0, 0, 0, 0, 0, 0]), ('DU', [0, 0, 0, 0, 0, 0
, 0]),
                                        ('CH', [0, 0, 0, 0, 0, 0, 0]), ('SIL', [0, 0, 0, 0, 0,
0, 0]),('Final Rank', [0, 0, 0, 0, 0, 0, 0])],
                                       orient = 'index', columns = ['KM', 'SL', 'PSC', 'DBS'])
```

Ranking Combination Methods

```
# Score Ranking
ranks = {}
points = [10, 8, 6, 4, 3, 2, 1]
algs = list(scoresTable.axes[1])

for i,method in enumerate(eval_methods):
    scores = sorted(list(zip(scoresTable.values[i:i+1, :].flatten().tolist(), algs)), key=lam
bda score: score[0], reverse=method[1])
    ranks[method[0]] = dict([(a[1], points[i]) for i,a in enumerate(scores)])

ranks

def sum_scores(alg):
    sum = 0
    for method in eval_methods:
        sum = sum + ranks[method[0]][alg]
    return (alg, sum)

scoreList = list(map(sum_scores, algs))
scoreList.sort(key=lambda tup: tup[1], reverse = True)

# Winner Ranking
def sum_wscores(alg):
    sum = 0
    for method in eval_methods:
        sum = sum + len(algs) -points.index(ranks[method[0]][alg])
    return (alg, sum)

winnerList = list(map(sum_wscores, algs))
winnerList.sort(key=lambda tup: tup[1], reverse = True)

sresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'KM']
wresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'KM']
sresult = sresult[0] +1
wresult = wresult[0] +1
fKM = (wresult + sresult) /2

sresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'SL']
wresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'SL']
sresult = sresult[0] +1
wresult = wresult[0] +1
fSL = (wresult + sresult) /2

sresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'PSC']
wresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'PSC']
sresult = sresult[0] +1
wresult = wresult[0] +1
fPSC = (wresult + sresult) /2

sresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'DBS']
wresult = [i for i, tupl in enumerate(winnerList) if tupl[0] == 'DBS']
sresult = sresult[0] +1
wresult = wresult[0] +1
fDBS = (wresult + sresult) /2

scoresTable.loc['Final Rank','KM'] = fKM
scoresTable.loc['Final Rank','SL'] = fSL
scoresTable.loc['Final Rank','PSC'] = fPSC
```

```
scoresTable.loc['Final Rank','DBS'] = fDBS
print(scoresTable)

if fKM == 1:
    walgo = 'KM'
elif fSL == 1:
    walgo = 'SL'
elif fPSC == 1:
    walgo = 'PSC'
elif fDBS == 1:
    walgo = 'DBS'

conn = sqlite3.connect("2MetaAttributes.db")
cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS Attr (id integer PRIMARY KEY, ma1 float, ma2 float, m
a3 float, ma4 float, ma5 float, ma6 float, ma7 float, ma8 float, ma9 float, walgo string)")
conn.commit()
# null is for autogenerated id value
cur.execute("INSERT INTO Attr VALUES (NULL,?,?,?,?,?,?,?,?,?,?)", (ma1, ma2, ma3, ma4, ma5, m
a6, ma7, ma8, ma9, walgo))
conn.commit()
```