ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
MSC IN DATA SCIENCE

Performance measures of clustering algorithms in retail industry

M.Sc. Thesis

Academic Supervisor:
Dimitrios Karlis

Student Name:
Ilias Liapikos

Business Supervisor:
Vasileios Georgiou

October 15, 2017

Contents

1 Introduction **4**

2 Clustering algorithms **6**

2.1 Cluster Analysis Applications 6

2.2 Clustering Algorithms Categorization 7

2.3 Representative clustering algorithms analysis 8

2.3.1 K-Means 8

2.3.2 Agglomerative Hierarchical 9

2.3.3 Model-based Clustering 10

2.3.4 DBSCAN 10

2.3.5 HDBSCAN 12

3 Clustering Validation **14**

3.1 Necessity of clustering result validation 14

3.2 Different approaches of validation 14

3.2.1 Relative Criteria 15

3.2.2 Internal Criteria 16

3.2.3 External Criteria 24

3.3 Statistical Validation 28

3.3.1 Data Resampling 29

3.3.2 Prediction Strength 29

3.3.3 Monte Carlo Simulations 30

4 Comparative Analysis **31**

4.1 Datasets Generation 31

4.2 Parameter Optimization 33

4.2.1 K-Means 34

4.2.2 Agglomerative Hierarchical 34

4.2.3 Gaussian Mixture Model 34

4.2.4 DBSCAN 37

4.2.5 HDBSCAN 40

4.3 Results 40

5 Product Data Application 44

5.1 Dataset Description 44

5.2 Procedure Description 45

5.2.1 Goal of application 45

5.2.2 Data objects’ distances 45

5.2.3 Validity index selection 46

5.2.4 Technique implementation 47

5.3 Results 49

5.3.1 General settings - Stop conditions 49

5.3.2 2-Dimensional Coordinates 49

5.3.3 40-Dimensional Coordinates 50

5.3.4 Conclusion - Further Improvements 54

Abstract

Cluster analysis (more often clustering) is a very powerful tool in a variety of fields: Statistics, social sciences, biology, machine learning, data mining and data reduction are significant representatives, with the latter being an example where clustering is not used as a stand-alone procedure but as a first step towards the goal. Despite the diversity of its applications, the core objective of clustering is to identify structures of similar objects inside vast datasets. The lack of knowledge of the exact result we try to identify, constitutes the part of clustering result validation the most crucial one.

In the current thesis, we present a brief description of the most used clustering algorithms along with the novelty they introduce to the procedure. We then make a strong effort to cover all the different approaches on the matter of *Clustering Validation* and how the nature of our problem defines the appropriate validity index. Following the theoretic approach, a comparative analysis in terms of result validation is implemented by an appliance of five different clustering algorithms in synthetic datasets. Finally, an approach on how clustering quality indices can be used in real product data is presented and evaluated.

Keywords: clustering,clustering algorithms,validity, indices,retail

1
Introduction

One could claim that **clustering** intuitively has its foundation on an inherent instinct of human nature: Categorize objects into groups that not only can be described by specific characteristics, but on the same time these same ”features” stand adequate enough to distinguish the underlying group from the remaining. In a more formal definition, *Cluster Analysis* groups data objects based on the information provided solely by features of the dataset. The principal goal, is to discover patterns among the vast volume of data that indicate significant negative or positive correlation among specific objects/actions.

The very attempt to give a formal definition of clustering reveals the ”weak” spots of the procedure:

- *The structure hypothesis*
  Right at the start of the the clustering procedure we hypothesize that there is indeed an underlying structure of the data under analysis, which we try to discover. This hypothesis may not stand in all cases.

- *The unsupervised nature of the problem*
  Even if a structure is existent in our dataset, previous to the analysis we only have a subjective opinion on what a valid clustering result would be. In a manner clustering techniques attempt to classify objects by assigning them cluster (class) labels, with the distinct specification that the labels are extracted from the dataset itself. For that reason, in scientific literature clustering may be referred to, as **unsupervised classification**.

- *The parameter selection*
  As stated on the definition, a clustering procedure attempts to reveal similar or proximate objects. Thus, it requires of us to define a measure of similarity or distance, which in turn will be used appropriately from the variety of clustering algorithms in order to produce the required result. The definition of the above can be quite challenging as it depends mostly on the kind of data processed. This delicate definition of a *proximity measure* is supplemented by a thorough choice of a *clustering criterion*. This criterion is expressed through a set of rules (more frequently in form of a cost function), and it depends mostly on the expected type of clusters we attempt to identify.

- *The result evaluation*
  Nested in all the above ”rough edges” of clustering, lies perhaps the most crucial issue: How good is our clustering result? Given the fact that we ”navigate in uncharted areas” there are many approaches on how to assess the validity of our results, depending on the nature of the data, the used algorithm and even the very expectation of what we are trying to accomplish through the clustering analysis.

Throughout this thesis we will present a detailed description of the different kind of clustering procedures, the algorithms that are most commonly used in modern day problems, as well as the types of validity measures that have been applied in recent studies. Furthermore, a comparative analysis of different clustering algorithms in synthetic data will be presented in order to highlight the respective strength and weaknesses of them when applied in data with specific structure. Finally, we will present an approach on how a clustering quality index can be used in real product data in order to create an hierarchical optimal clustering based on a different feature at each level.

2
Clustering algorithms

2.1 Cluster Analysis Applications

Clustering as a procedure, has been proven useful throughout scientific research not only in divergent disciplines but in fundamentally different ways.

Perhaps, the most recent realization of clustering usefulness is that of *data reduction*. In modern day problems the amount of available data constitutes an analysis on the whole dataset nearly impossible. Thus, clustering comes to perform an initial partition of the dataset and enable analysis through cluster representatives. On the contrary, the oldest utility of clustering (perhaps even the spark to have lit the flame of the procedure expansion) is that of *data understanding*.This is either expressed in terms of identifying a taxonomy in nature or human populations, or the detection of disease symptoms patterns and sensitive sub-populations.Another major field where clustering plays a pivot role is that of business. Customer data constitute a huge source of information from where clustering techniques can extract sale patterns, seasonal fluctuations, and customer behavioural characteristics. All these in turn, can boost targeted marketing campaigns and help identify business opportunities. Finally, we have to point out that a very popular field of use for clustering techniques is that of *fraud detection.* It has been used as an initial step to identification of fraudulent actions or users because of its ability to, relatively easy, point out extreme cases in the dataset by assigning them in the same cluster, way far from the remaining ones.

2.2 Clustering Algorithms Categorization

Complete versus Partial

A self-explanatory categorization of the clustering algorithms arises from whether all data points (objects) are assigned to a cluster or not. In an era where data is collected in massive amounts and extreme pace, it is immediate that a significant part of it does not belong to well-defined groups. Thus, it is quite useful for an algorithm to allow a percentage of unassigned objects that depending to the problem may represent noise, extreme cases or even just low informative representatives of the dataset.

Crisp versus Fuzzy

With the term *Crisp Clustering* we refer to clustering techniques that assign each object in a single cluster. In the opposing side *Fuzzy Clustering* allows each object to belong in every cluster discovered by the algorithm denoting the power of affiliation with them in terms of a weight vector. The weight values vary from zero (certainty of non-belonging) to one (certainty of belonging), and it is not necessary to sum to one. A very similar approach to fuzzy clustering is followed on *probabilistic clustering* where the weights are replaced by probabilities to belong to each cluster.
The latter techniques are quite useful to avoid clustering in an arbitrary way objects that are quite ”close” in more than one clusters. Finally both these, techniques can be transformed to crisp clustering simply by assigning the object under examination to the cluster with the highest weight/probability.

Cluster formulation variations

One of the core characteristics that represent the operation of a clustering algorithm is the way that it defines clusters and operates to recognize them. The most common variations sourced in that choice are the following:

- *Partitional*
  This kind of algorithms identify a set of disjoint clusters. This is achieved by applying a partition of the data set, producing an integer number of sub-spaces each depicting a different cluster. The procedure is iterative and works towards optimizing a set of criteria.

- *Hierarchical*
  The fundamental difference of *hierarchical clustering* compared to partitional is that each cluster is realised as a union of specific sub-clusters. The whole dataset is depicted in a tree form with the root to be a huge cluster containing all data objects while the leaf nodes are singleton clusters containing single data objects. This hierarchy can be formed either ”bottom-up” (*aglomerative approach*) or ”top-down” (*divisive approach*).

- *Density Based*
  In this type of algorithms the clusters are defined as regions of high density, surrounded by areas of low density. They introduce a flexibility on the shape of the cluster identified, while on the same time can identify noise or extreme cases on the dataset.

- *Model Based*
  In model based clustering clusters are realized as mixture components of a finite mixture of distributions.

2.3 Representative clustering algorithms analysis

As we have made clear so far a variety of approaches exist for each clustering problem. In order to highlight the differences of each approach along with the benefits that come along it we will describe in detail the algorithms that were used throughout the thesis.

2.3.1 K-Means

This is perhaps the most cited and commonly used clustering algorithm. It performs a partitional,complete,crisp clustering. The metric used to determine the distance of data objects is **Euclidean distance**, while the goal is to minimize the following objective function:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i),$$

where C_i is the i-th cluster and $d(x, \mu_i)$ is the Euclidean distance between each data object and the center of it. As a representative of the divisive clustering algorithms its operation is iterative and its main algorithmic steps are the following:

1. Select k cluster centers (either randomly or explicitly).
2. Calculate the Euclidean distance between all points and all cluster centers.
3. Assign each data point to the cluster, from whose center it has the minimum distance.
4. Recalculate the new cluster centers with the use of

   $$\frac{1}{|C_i|} \sum_{x \in C_i} d(x, \mu_i),$$

   where $|C_i|$ is the cardinality of the i-th cluster.

5. Repeat steps 2-4 iteratively until there is no change on the cluster centers or a minimum user-set threshold of center shift occurs.

The main dis-functionality of this algorithm is that we have to predetermine the number of the clusters which are unknown in an unsupervised problem as clustering. Moreover, the initialization of the hypothesized cluster centers can affect the resulting clustering crucially. In practice, either we choose very distant data points or perform a multiple random initial assignments in order to observe a dominating clustering result. On the other hand, this is a computationally light procedure and in case the data are well separated produces an excellent result.

2.3.2 Agglomerative Hierarchical

The two key parameters that characterize an hierarchical clustering algorithm are its **distance metric** and **linkage criterion**. The scope of our analysis was met by an implementation of an agglomerative hierarchical algorithm using as metric the *Euclidean distance* and as linkage criterion the *Ward's criterion.*

In more details, the algorithm starts with every data object as a singleton clusters and moves its way up by merging in each step the pairs of clusters that minimize the within-cluster variance of the newly merged-to be cluster. In a strictly mathematical expression (and denoting the Ward2 implementation we followed in the following chapters) the cluster update formula is written as following:

$$\delta(i \cup i') = \left[\frac{w_i + w''_i}{w_+ w'_i + w''_i} \delta^2(i, i'') + \frac{w'_i + w''_i}{w_+ w'_i + w''_i} \delta^2(i', i'') - \frac{w''_i}{w_+ w'_i + w''_i} \delta^2(i, i') \right]^{1/2}$$

$$\text{and} \quad w_{i \cup 1'} = w_i + w'_i,$$

where w_i is the cardinality of cluster i and $\delta^2(i, i') = \sum_j (x_{ij}, x_{i'j})^2$, meaning that the input dissimilarities are expressed as *squared Euclidean distance*. Taking the resulting hierarchy and visualizing it as a tree allows the user to cut at the desired height and obtain a complete,crisp clustering result. This creates a certain amount of bias as selecting different heights can lead to different number of clusters which inevitably will have to choose as a better fitted in our needs result.

2.3.3 Model-based Clustering

Model based clustering uses the hypothesis that the d-dimensional features of a data object are a sample of a finite mixture density

$$p(\mathbf{x_i}|K,\theta_K) = \sum_{k=1}^{K} p_k \phi(\mathbf{x_i}|\mathbf{a_k}),$$

where p_k's stand as the mixing proportions, $\phi(\cdot|\mathbf{a_k})$ denotes the distribution density and θ_K is the parameter vector.

The most common mixtures used is that of *Multivariate Gaussian mixtures*. In that case the distribution $\phi(\cdot|\mathbf{a_k})$ takes the form:

$$\frac{1}{\sqrt{2\pi|\Sigma_k^{-1}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1} (\mathbf{x}-\mu_x) \right),$$

where $\mathbf{x} = (\mathbf{x_1}, \cdots, \mathbf{x_n})$ is the vector of the n d-dimensional data objects μ_k is the vector of means $(\mu_1, \cdots, \mu_k)$, Σ_k is the $d \times d$ covariance matrix of class k respectively.

The standard methodology of model-based clustering includes an implementation of the *Expectation-Maximization algorithm* to estimate the finite mixture model components and using the *Bayesian Information Criterion* to select the number of mixture components. This methodology has been criticized in terms of efficiency to estimate the correct number of clusters in the dataset and the various ways to to bypass this are discussed in further section of the thesis.

2.3.4 DBSCAN

The acronym *DBSCAN* stands for Density Based Spatial Clustering for Applications with Noise and was introduced in 1996 by *Martin Ester et. al.* It brought a revolutionary approach on the table of clustering algorithms as it abandoned the formulation of cluster in a nested or non-nested way as the hierarchical and divisive clustering algorithms perceive them respectively. The general goal of DBSCAN and all density-based algorithms extensively is to locate data space regions of high density surrounded by regions of low density, without having to predetermine how many clusters you expect to find (e.g. K-means) or how to terminate the division or merging of nested clusters (e.g. Hierarchical clustering). In that manner density based algorithms can handle noise and discover non-convex shaped clusters but unfortunately in order to define what constitutes a *dense region* or not they introduced a couple of extra parameters that affect the resulting clustering in a crucial way.

In order to explain how DBSCAN works we have to introduce the reader to some basic definitions:

Eps Neighbourhood: The eps neighbourhood of a point p is defined by:

$$N_{Eps}(p) = q \in D | d(p,q) \leq Eps.$$

Direct Density Reachability: A point p is *directly density-reachable* from a point q with respect to Eps,MinPts if:

1. $p \in N_{Eps}(q)$
2. $|N_{Eps}(q)| \geq MinPts$

Density Reachability: A point p is *density-reachable* from a point q if there is a chain of points $p_1, p_2, \dots, p_n$, where $p_1 = p, p_n = q$ such that p_{i+1} is directly density-reachable from p_i.

Given the aforementioned definitions, all the data points (objects) can be separated in three distinct categories:

1. **Core points**: A point is characterized as *core point* if in its neighbourhood (defined by a distance metric and the user-defined parameter Eps), lie more than MinPts (again a user-defined parameter) points.
2. **Border points**: In turn, as *border point* is defined every point that is not a core point but lies in the neighbourhood of one.
3. **Noise points**: *Noise points* are defined to be all the remaining points of the dataset that do not fall in the above categories.

Combining the above-mentioned definitions we can describe conceptually the procedure of the DBSCAN clustering:

- All data points are labeled as core,border and noise points according to the input parameters Eps,MinPts.
- By definition all core points belong to a cluster.
- All core points that are directly density -reachable from another core point belong to the same cluster.
- All border points that are density-reachable from a core point belong to the cluster of the core point. In case a border point is density-reachable from non-direct density reachable core points, the cluster assignment need to be resolved automatically.
- All noise points remained unassigned.

2.3.5 HDBSCAN

The final clustering algorithm we used throughout this thesis is a combination of hierarchical and density-based algorithm: *Hierarchical-DBSCAN (HDBSCAN).*

The first crucial difference from DBSCAN is the metric on which density of the data space is evaluated. The notion, is that we want to enhance even more the difference between dense and non-dense regions and this is accomplished by introducing the **mutual reachability distance** metric:

$$d_{mreach}(\mathbf{x}_p, \mathbf{x}_q) = max\{core_k(\mathbf{x_p}), core_k(\mathbf{x_q}), d(\mathbf{x_p}, \mathbf{x_q})\},$$

where $d(\mathbf{x_p}, \mathbf{x_q})$ is the original metric distance between the two objects, while $core_k(\cdot)$ is the distance to the k-nearest neighbour. It is immediate from that metric, that points belonging in dense regions of the data space (meaning low core distance) retain their original distance while points in low-density areas are driven further apart.

After the computation of the *mutual reachability distance* between all pairs of data, there is a need to distinguish in practice the more dense regions. This is achieved by perceiving our dataset as a graph where data point are represented by vertices and the edges connecting them have a weight equal to the mutual reachability distance between them. Using that graph the Minimum Spanning tree can be formed by adding each time the lowest weight edge that connects the already formed tree to a vertex not yet in it (Prim's algorithm).After that the MST is extended by adding an edge for every vertex pointing to each self with the weight of the core distance of that object. Finally if iteratively all edges in a decreasing weight are removed we can obtain the *HDBSCAN hierarchy.*

Note:Up until this point the procedure can be described as equivalent to DBSCAN but with the use of another distance metric. In fact, if we use the minPts parameter of DBSCAN as the k in k-nearest neighbours we will have a clustering tree with all the different partitions that we can obtain for $Eps \in [0, +\infty]$. In other words Eps would be a threshold above which all edges of the extended MST would be removed and the remaining connected components would represent the found clusters (all singleton points would be considered noise).

At this very point HDBSCAN comes to surpass DBSCAN and get rid of the unintuitive ,crucial to DBSCAN, Eps parameter.The notion of this part of the algorithm is to transform the hierarchical density tree to a more compact form that would help uncover the most persistent clusters in our dataset. This is accomplished with the use of the (usually) only parameter of the algorithm: the **minimum cluster size**. Using that parameter we revisit the hierarchy tree and at every split we examine if the produced clusters meet the minimum size. In case at least on of them does not, we consider the split undone and retain the cluster label of the parent while we mark which points "fell off" the parent cluster and at what distance. This produces a much more simple hierarchy and reduces the size of the three drastically.It is useful to notice that in most implementations of HDBSCAN algorithm the minimum cluster size coincides with the k in k-neighbours used for the definition of the mutual reachability distance, but it is not necessary.

At the final stage of the algorithm we want to extract the most dominant clusters of the dataset in order to obtain a "flat" clustering. In real datasets though, these clusters may vary in density which is something that cannot be detected with the use of a global density level like in DBSCAN. Thus, HDBSCAN defines a different measure that will allow a variety in density of the final chosen clusters,

Cluster Stability: The stability of a cluster C_i is given by

$$S(C_i) = \sum_{\mathbf{x_j} \in C_i} \Big(\lambda_{max}(\mathbf{x}_j, C_i) - \lambda_{min}(C_i) \Big),$$

where $\lambda_{min}(C_i)$, is the minimum density level where C_i starts to exist and $\lambda_{max}(\mathbf{x}_j, C_i)$ is the maximum density level after which object $\mathbf{x}_j$ does not belong to that cluster.

With the use of *cluster stability* the goal of obtaining a flat clustering of the most stable clusters is reduced to the following optimization problem:

If $\{C_2, \ldots, C_k\}$ is the collection of all clusters in the condensed hierarchical tree except the root C_1 we wish to maximize the sum of stabilities of the extracted clusters

$$\max_{\delta_1, \delta_2, \ldots, \delta_k} \quad J = \sum_{i=2}^{k} \delta_i S(C_i),$$

$$\text{with respect to} \quad \begin{cases} \delta_i \in \{0,1\}, i = 2, \ldots, k \\ \\ \sum\limits_{j \in \mathbf{I}_h} \delta_i = 1, \forall h \in \mathbf{L} \end{cases},$$

where δ_i indicates if cluster i is included on the flat solution $(\delta_i = 1)$ or not $(\delta_i = 1)$,$\mathbf{I}_h$ is the set of indexes of leaf nodes and $\mathbf{L}$ is the set of indexes of all clusters in the path the respective lead node to the root.

This optimization problem is solved by selecting initially all the leaf nodes as part of the flat clustering move upwards the hierarchy, including a parent cluster in the solution (and simultaneously erasing all its descendants from it) only if its stability is greater than the sum of the stabilities of its descendants. Once we reach the root node we call the set of currently selected clusters the solution to the optimization problem and return the crisp clustering result.

3
Clustering Validation

3.1 Necessity of clustering result validation

As it was briefly described in the introduction clustering problems inherit some difficulties from its very conceptual formulation. It is not only the ambiguity on whether there exists an underlying structure on our dataset or not, but question about matters such as ”optimal number of clusters”, ”optimal clustering algorithm”, ”business value of the resulting clustering” rise all the time from the researchers comparing and applying clustering techniques. Moreover, it is quite common to apply the same clustering algorithm with various sets of input parameters and obtain quite different results. All the above highlight the important to define universally a way to assess the clustering result.

3.2 Different approaches of validation

Being able to determine the existence of a non-random structure of the dataset, the optimal number of clusters and in general compare two clustering results against each other are the hopeful outcomes of the validation stage of clustering.

The core difference between the various proposed measures, referred commonly as **validity measures**, lies on whether the information they use to define the clustering quality has its source internally or externally of the dataset. Based on that discrimination *validity criteria* are classified as:

- Internal criteria

  Indices using quantities and features available within the dataset.

- External criteria

  Quality indices that compare the resulting clustering to a ground-based truth available externally either by previous research or by subjective knowledge from field experts.

Finally, usage of indices (either internal or external) to perform a comparison between clustering results obtained from the same clustering algorithm but differently parametrized, is sometimes referred in bibliography as a separate category under the name **relative criteria**.

3.2.1 Relative Criteria

The basis of cluster validity with the use of relative criteria lies on the comparison of different clustering results and the selection of the best in terms of a predefined criterion. It is usually applied when trying to evaluate results of a specific clustering algorithm with different parametrization and can differ in the following way:

- The set of varying parameters does not involve the number of clusters.

  In that case the clustering algorithm is applied with a broad spectrum of varying parameters. From the obtained results the largest range of parameters for which the identified number of clusters remains steady is chosen. The set of parameters that yields the best result is then chosen to be the middle of that range respectively. In that manner we also, indirectly determine the underlying number of clusters in our dataset.

- The set of varying parameters involves the number of clusters.

  This is the most common case of use and it based on a validity index, q. The procedure is described as following:

  1. Predetermine the range of acceptable values for the number of clusters parameter. e.g. $M \in [M_{min}, M_{max}]$
  2. Run the clustering algorithm for each of the values in the above interval using different set of remaining initial parameters.
  3. Determine for each set of runs the set of parameters that return the best clustering scheme in terms of the specified validity index q.
  4. Plot the best value of the validity index for each number of clusters parameter versus the number of clusters.

   (a) In case the validity index does not behave in a monotonic way as the number of clusters parameter increases we seek for the local optima depending on the choice of index.
   (b) In case the validity index do behave in a monotonic way we search for the number of cluster value where a significant local change of the validity index occurs (a knee in the plot). In case it is absent we may receive it as an indication that there is no underlying structure on our data.

5. After establishing the optimum value for the number of clusters parameter we can supplement it by the set of the remaining parameters that yielded that result and are previously stored.

3.2.2 Internal Criteria

Most of the internal criteria used for cluster validity purposes, are based on the notion that a good clustering result should include the following characteristics:

- Compactness

  The resulting clusters should be homogeneous. This means that members of the same cluster should be as close to each other (in terms of a specified distance metric) as possible.

- Seperation

  The resulting clusters should be widely spread in the data space. The distance between clusters can be measured with either of the following:

  (a) Single Linkage
  Inter-cluster distance is measured between the closest members of the cluster.

  (b) Complete Linkage
  Inter-cluster distance is measured between the furthest members of the cluster.

  (c) Representative Comparison
  Inter-cluster distance is measured between the representatives of each cluster. e.g. its centroids

- Connectedness

  Neighbouring points should belong to the same cluster

Compactness & Separation - based criteria

A group of internal validity measures based, directly or indirectly, on these upper-described notions are the following:

- Sum of Squares Within Clusters (SSB_K)

  $$SSW_K = \sum_{i=1}^{K} \sum_{x \in C_i} d(x - c_i)$$

  This is a widely used measure of cluster *cohesion* in case of Euclidean distance.

- Sum of Squares Between Clusters (SSB_K)

  $$SSB_K = \sum_{i=1}^{K} |C_i| d(c_i - c)$$

  This is a widely used measure of cluster *separation* in case of Euclidean distance.The higher the value of the index the more separated the resulting clusters are from another.

- Calinski & Harabasz (CH)

  $$CH = \frac{\frac{SSB_K}{(K-1)}}{\frac{SSW_K}{(N-K)}}$$

- Ball & Hall (BH)

  This index is set to be the mean,through all K clusters, of the cluster's mean dispersion. In case of clusters of equal cardinality it is given via:

  $$BH = \frac{SSW_K}{N}$$

- Dunn's Index $(Dunn)$

  We define the minimal distance between data objects that do not belong to the same cluster:

  $$d_{min} = \min_{i,j=1, i \neq j}^{K} \min_{x \in C_i, x' \in C_j} d(x, x').$$

  Moreover we define the largest distance between points of the same cluster, also known as *cluster diameter*:

  $$diam(C_k) = \max_{x, x' \in C_k} d(x, x').$$

  Based on these quantities *Dunn's index* is calculated as:

  $$Dunn = \frac{d_{min}}{\max_{k=1}^{K} diam(C_k)}$$

- Davies & Bouldin (DB)

  DB index is based on the similarity measure R_{ij}. Let us define the mean distance of the points belonging in cluster i:

  $$s_i = \frac{1}{|C_i|} \sum_{x \in C_i} d(x - c_i),$$

  and the distance between the cluster's geometrical means

  $$d_{ij} = d(c_i, c_j).$$

  We then can define the similarity index as following:

  $$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

  , while $R_i = \max_{i,j=1,\dots,K} R_{ij}, i \neq j$. The *Davies & Bouldin* index is then evaluated as:

  $$DB = \frac{1}{K} \sum_{i=1}^{K} R_i.$$

  Conceptually the DB index represents the average similarity between each cluster and its most similar one. Thus, it is desirable to have the minimum possible average similarity of the clustering scheme.

- Scattering-Separation index (SD)

  The SD index is based on the concepts of the *average cluster scattering* and the *total separation between clusters*. Let us define these two concepts via the following formulas:

  $$Scatt = \frac{1}{K} \sum_{k=1}^{K} \frac{\|V_k\|}{\|V\|} \quad , \quad Dis = \frac{D_{max}}{D_{min}} \sum_{i=1}^{K} \frac{1}{\sum_{j=1,\, j \neq i}^{K} d_{ij}},$$

  where $\|V_k\|$ is the respective cluster variance, $\|V\|$ is the dataset variance, d_{ij} is the distance between the cluster centers as defined in the DB index, while D_{max} & D_{min} are the respective maximum and minimum values of that distance.

  The SD index is defined as

  $$SD = c \cdot Scatt + Dis,$$

  where c is a weighting factor corresponding to the total separation value of the resulting partition with the greatest number of clusters (thus, it has to be calculated first). Well defined clusters are considered to have small scattering and great separation amongst them which leads to the fact that we search for clusterings with small values of the SD index.

- Silhouette index (SC)

  The Silhouette value of a data object in contrast to the above defined indices, measures the degree of confidence in its clustering assignment:

  $$S_i = \frac{b_i - a_i}{max\{a_i, b_i\}},$$

  where a_i denotes the average distance between point i and the points assigned to the same cluster and b_i the average distance between itself and the points assigned to the "nearest" neighbouring cluster

  $$a_i = \frac{1}{|C_i|}\sum_{j\in C_i} d(i,j), \qquad b_i = \min_{C_k, k\neq i}\sum_{j\in C_k}\frac{d(i,j)}{|C_k|}.$$

  Let us set as the *Silhouette index* of a clustering scheme, the average Silhouette score of its objects:

  $$SC = \frac{1}{N}\sum_{i=1}^{N} S_i.$$

  The values of the index lie in $[-1,1]$ with unity denoting a perfect assignment. Thus, in practice values around 0.5 and higher are considered acceptable.

- Connectivity ($Conn$)

  Let $nn_{i(j)}$ denote the j-th nearest neighbour of observation i and

  $$x_{i,nn_{i(j)}} = \begin{cases} 0, & when \quad i,j \quad \text{belong to the same cluster} \\ \frac{1}{j}, & otherwise \end{cases} .$$

  Then for particular clustering partition the *Connectivity index* is obtained through:

  $$Conn = \sum_{i=1}^{N}\sum_{j=1}^{L} x_{i,nn_{i(j)}}.$$

Hierarchical clustering criteria

Most of the above-defined indices are meaningful to be used in partitional clustering. At this point we will define four indices that must be used simultaneously and sequentially at each step of an hierarchical clustering to evaluate the correct number of cluster in the dataset. Finally we will present an index that can evaluate solely a clustering hierarchy.

- Root Mean Squared Standard Deviation ($RMSSTD$)

  $RMSSTD$ measures the homogeneity of the formed clusters at each step by calculating the square root of the pooled sample variance of all features. Thus in each step it should decrease, otherwise we are moving towards a less homogeneous solution. It is defined as following:

$$\sqrt{\frac{\sum_{i=1}^{K}\sum_{j=1}^{d}\sum_{m=1}^{n_{ij}}(x_m-\bar{x}_j)^2}{\sum_{i=1}^{K}\sum_{j=1}^{d}(n_{ij}-1)}},$$

  where d is the dimension of the data objects (number of features),n_j is the number of data values of j-th dimension, n_{ij} is the respective number of data values assigned to i-th cluster, while $\bar{x}_j$ is the mean of data values of j-dimension.

- R-squared (RS)

  RS measures degree of difference between clusters. It is defined as:

$$RS = \frac{SSB_K}{SSB_K + SSW_K}.$$

  It is obvious from the definition that the greater the differences between the clusters, the more homogeneous are the participant clusters. Thus, we seek for values of the index more close to the right border of the range interval $\big([0,1]\big)$.

- Cluster Distance (DC)

  The CD index calculates the distance between the two clusters that are merged (or of the resulting clusters of a division) according to the linkage function used for the produce of the hierarchy. Obviously, the resulting clusters should be as further as possible.

- Semi-Partial R-squared (SPR)

  This index measures the loss of homogeneity after merging two clusters (or equivalently dividing a single cluster to two). Its range is $\big([0,1]\big)$, with 0 standing for the merging two completely homogeneous clusters. Thus, in an agglomerative approach we seek for index values more close to unity. The index is defined as:

$$\frac{SSW^{C_1 \cup C_2} - SSW^{C_1} - SSW^{C_2}}{SSW_K + SSB_K}.$$

  These four indices are calculated at each step of the hierarchical algorithm and plotted against the number of clusters. The occurrence of a knee indicates that there is an underlying structure on the dataset and we have isolated the optimal number of clusters.

Correlation Criteria

This particular family of indices differs a lot in terms of how it handles the internal information of the data to validate the result. It is based in terms of correlation between matrices or vectors depicting in varying ways the inherent dataset information.

- Cophenetic Correlation Coefficient (*CPCC*)

  The core tool of this index (subsequently limiting down its use only to hierarchical clustering results) is the **Cophenetic distance** e.g. *The distance between two data objects at which an aglomerative hierarchical algorithm assigns them on the same cluster for the first time.*

  Let b_{ij} the elements of the original *proximity matrix P*, and c_{ij} the elements of the produced *cophenetic matrix* P_C. We define the *CPCC* index as:

$$CPPC = \frac{N_T^{-1}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(b_{ij}\cdot c_{ij}-\mu_p\cdot\mu_c)}{\sqrt{\left[N_T^{-1}\cdot\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(b_{ij}^2-\mu_p^2)\right]\left[N_T^{-1}\cdot\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(c_{ij}^2-\mu_c^2)\right]}},$$

  where $N_T = \frac{N\cdot(N-1)}{2}$ is the total number of pairs of distinct point in the dataset and μ_p,μ_c are the mean vectors of the two matrices. e.g.

$$\mu_p = N_T^{-1}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}P(i,j) \quad , \quad \mu_c = N_T^{-1}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}P_C(i,j).$$

  This index approaches the notion of clustering validation in terms of measuring the correlation between the original proximity matrix used for the clustering procedure and another matrix carrying internal or external information about the resulting clustering. In that manner it can be used as a notion into external indices as we will observe on the next section. Finally as a correlation measure indicates significant similarity between the two matrices (meaning a well fitted clustering to the data) for values approximate to unity.

- Gamma index

  The Gamma index of Baker-Hubert [1975] is an adaptation of the rank correlation measure of Goodman & Kruskall for clustering purposes. It was originally proposed to determine the optimal number of clusters in hierarchical clustering, but modern packages have extended its use in every clustering result.

  The notion behind this index is to perform a comparison of the resulting within-cluster dissimilarities and the respective between-cluster dissimilarities. This achieved by evaluating the correlation of two vectors; one carrying internal information of the dataset and the other representing the clustering result. In more details the first vector contains the distances of pairs of data objects while the latter is a matrix taking binary values, with one depicting data pairs that belong to the same cluster while zero denotes pairs of data belonging in different clusters (always based on the clustering result). The index then is calculated as:

$$\Gamma = \frac{s^+ - s^-}{s^+ + s^-},$$

where s^+ represents the number of times where two points belonging in the same cluster had strictly smaller distance than two points that are classified into different clusters (*concordant pairs*) while s^- denotes the reverse count (*discordant pairs.* The usual implementations of Γ index do not take account the ties in the above computations.

Both vectors have length N_T (the number of distinct pairs in the dataset. Let us denote the number of distinct pairs inside a cluster as:

$$N_W = \sum_{k=1}^{K} \frac{n_k(n_k - 1)}{2}.$$

It is apparent that the number of distinct pairs of object that do not belong to the same cluster is

$$N_B = N_T - N_W.$$

Thus, the number of comparison required for the calculation of the index is $N_W \cdot N_B$. This constitutes the computation of the index quite expensive in case of large datasets,compared to the rest of the internal indices.

- $G+$ *index*

  This index is a simple variation of the Γ index. It is calculated as the proportion of the discordant pairs of data objects within the dataset:

  $$\frac{s^-}{\frac{N_T(N_T-1)}{2}}.$$

- Tau index

  Another variation of the Γ *index* is:

  $$\tau = \frac{s^+ - s^-}{\frac{N_T(N_T-1)}{2}}.$$

This index has the advantage of being able to take a modified form that also takes into account the ties among the comparisons performed for the numerator:

$$\tau_c = \frac{s^+ - s^-}{\sqrt{(v_0 - v_1)(v_0 - v_2)}},$$

where $v_0 = \frac{N_T(N_T-1)}{2}, v_1 = \sum_i \frac{t_i(t_i-1)}{2}, v_1 = \sum_j \frac{u_j(u_j-1)}{2}$,
t_i is the number of ties in the i-th group of comparisons for the vector of distances, while u_j is the respective sum for the binary vector of the clustering result. By taking advantage of the binary form of the latter vector it is easily proven that $(v_0 - v_2) = N_W \cdot N_B$. Finally making a last (quite plausible) hypothesis that there are a few identical entries in the vector of distances the following modified *tau index* formula is computed:

$$\tau_c = \frac{s^+ - s^-}{\sqrt{v_0(N_W \cdot N_B)}}.$$

Stability Criteria

A final sub-category of internal indices are the *stability indices.*Their distinctive characteristic is that they compare results from clustering based on the whole dataset to clustering based on the dataset after the universal removal of one feature at a time. This is extremely useful in clustering analysis of highly correlated data or in general data with an extensive amount of features in comparison to the number of observations $(p >> n)$.

- Average Proportion of non-overlap (APN)

  The *APN* index measures the average proportion of data objects that are not assigned to the same cluster based to the full data versus the data with the removal of one feature at a time. It is defined through:

  $$APN = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{l=1}^{M} \left(1 - \frac{|C^{i,l} \cap C^{i,full}|}{|C^{i,full}|}\right),$$

  where $C^{i,full}$ represent the cluster on which object i is assigned when clustering is based on the whole dataset and $C^{i,l}$ is the respective cluster based on the dataset with the removal of the l-th feature. The index values vary between zero and unity, with zero denoting highly consistent clustering results.

- Average Distance (AD)

  The *AD* index measures the average distance between data objects assigned in the same cluster through full and partial clusterings. It is defined as :

  $$AD = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{l=1}^{M} \frac{1}{|C^{i,full}| \cdot |C^{i,l}|} \left[\sum_{i \in C^{i,full},\, j \in C^{i,l}} d(i,j) \right].$$

The index lies in the interval $[0, \infty]$, and we seek small values of it.

- Average Distance between means (ADM)

  The ADM is defined respectively to AD but instead of measuring the average distance of all cluster participants, it calculates the average distance of the cluster centers.It lies, also in the same interval and in general we wish to keep it minimal.

- Figure of merit (FOM)

  The FOM index measures the intra-cluster variance of the data objects in the deleted-feature when the clustering is based on the remaining ones. For the respective l-th feature the FOM index is defined as:

$$FOM_l = \sqrt{\frac{1}{N}\sum_{k=1}^{K}\sum_{i\in C_k(l)} d(x_{i,l}, \overline{x}_{C_k(l)})},$$

  where $x_{i,l}$ is the value of observation i in the deleted feature and $\overline{x}_{C_k(l)}$ is the average of cluster $C_k(l)$.

  Each column-wise FOM index is multiplied by a factor of $\sqrt{\frac{N}{N-K}}$ in order to fade out the tendency to decrease as the number of clusters K increases. Finally, the final FOM index is calculated to be the average between all column-wise scores and varies between zero and ∞. Smaller values indicate better clustering results.

3.2.3 External Criteria

The usage of external indices to validate the clustering results occurs only when there is a prior knowledge of the underlying structure of the data which we wish to evaluate or when indeed the general truth of the dataset is given and we want to obtain the best clustering approach to accomplish it. For the extend of this sub-section we will refer to the external information as partition of the dataset to classes while the clustering result as the identified clusters.

”Classification” Criteria

The following two measures ”borrow” their perspective from classification measures.

- Entropy ($Entrp$)

  This index attempts to measure the degree on which each cluster consists of data objects of the same class. In particular the entropy of cluster C_k is defined as:

$$Entrp(C_k) = \frac{-1}{\log q}\sum_{i=1}^{q}\frac{|C_k^i|}{|C_k|}\log\frac{|C_k^i|}{|C_k|},$$

Internal indices			
Index	**Abbreviation**	**Range**	**Rule**
Sum of Squares Within Clusters	SSB_K	$[0,+\infty]$	min
Sum of Squares Between Clusters	SSW_K	$[0,+\infty]$	max
Calinski-Harabasz	CH	$[0,+\infty]$	max
Ball-Hall	BH	$[0,+\infty]$	max diff
Dunn	CH	$[0,+\infty]$	max
Davies-Bouldin	DB	$[0,+\infty]$	min
Scattering-Separation	SD	$[0,+\infty]$	min
Silhouette	SC	$[-1,1]$	max
Connectivity	$Conn$	$[0,+\infty]$	min
Cophenetic Correlation	$CPPC$	$[-1,1]$	max
Gamma	Γ	$[-1,1]$	max
Gamma plus	$\Gamma+$	$[0,1]$	min
Tau	τ_c	$[-1,1]$	max
Average non-overlap proportion	APN	$[0,1]$	min
Average Distance	AD	$[0,\infty]$	min
Average Distance between means	ADM	$[0,\infty]$	min
Figure of Merit	FOM	$[0,\infty]$	min

Table 3.1: Identify good clustering results via internal indices

where q is the number of distinct classes in the partition and $|C_k^i|$ denotes the number of data objects of class i assigned in cluster k. Finally the *entropy* of the clustering schema is defined as:

$$Entrp = \sum_{k=1}^{K} \frac{|C_k|}{N} \cdot Entrp(C_k).$$

A perfect clustering would result to clusters containing objects from a sole class, leading the entropy of the schema to zero. In general, small values of the index are required.

- Purity (Pur)

  Purity measures the extend to which each cluster contains data objects from primarily one class.In particular, the purity of cluster C_k is defined as:

$$Pur(C_k) = \frac{1}{|C_k| \max_i(|C_k^i|)},$$

where the overall purity of the clustering schema is:

$$Pur = \sum_{k=1}^{K} \frac{|C_k|}{N} \cdot Pur(C_k).$$

Respectively a perfect clustering would result to clusters containing objects from a sole class, leading the purity of the schema to unity. In general, big values of the index are required.

Pair-Counting Criteria

Let us define based on the external dataset partition P and the clustering schema C, the following outcomes:

(a) Two data objects belong to the same group according to both P & C.

(b) Two data objects belong to the same group according to P but not according to C.

(c) Two data objects belong to the same group according to C but not according to P.

(d) Two data objects do not belong to the same group according to both P & C.

If we denote the number of pairs contained in each category as yy, yn, ny, nn, we have that the total distinct pairs of objects of the dataset N_T is equal to the sum of the above. Based on these notions and their derived confusion matrix of the two schemas we define the following external indices:

- Precision $(Prec)$

  Using as reference the partition P this index measures the proportion of data objects rightly grouped in the clustering schema:

  $$Prec = \frac{yy}{yy + ny}.$$

- Recall (Rec)

  This index measures the proportion of data objects grouped together at P which are also grouped together in C:

  $$Rec = \frac{yy}{yy + yn}.$$

- Czekanowski-Dice or Ochiai (CD)

  This index is the harmonic mean of the *precision* and *recall* indices, making it identical to the *F-measure* as defined in classification problems:

  $$CD = \frac{2 \cdot yy}{2 \cdot yy + yn + ny}.$$

- Kulczynski (KU)

  This index is the arithmetic mean of the *precision* and *recall* indices:

$$KU = \frac{1}{2}\left(\frac{yy}{yy+ny} + \frac{yy}{yy+yn}\right).$$

- Fowlkes-Mallows (FM)

  This index is the geometric mean of the *precision* and *recall* indices:

$$FM = \frac{yy}{\sqrt{(yy+yn)\cdot(yy+ny)}}.$$

- Jaccard Coefficient (J)

$$J = \frac{yy}{yy+yn+ny}.$$

- Rand Index (RI)

$$RI = \frac{yy+nn}{N_T}$$

- Adjusted Rand Index (ARI)

  Rand index has been identified to suffer from some limitations, such as that while increasing the number of clusters it approaches unity even when comparing independent clusterings. Furthermore, it has a non-constant expected value. In order to bypass these problems *Hubert & Arabie* introduced the *Adjusted Rand index*: A null hypothesis is made, assuming that the index follows a generalized hypergeometric distribution and *ARI* is defined as the difference of *Rand index* and its expected value under the null hypothesis

$$ARI = \frac{N_T\cdot(yy+nn) - [(yy+yn)(yy+ny)+(ny+nn)(yn+nn)]}{N_T^2 - [(yy+yn)(yy+ny)+(ny+nn)(yn+nn)]}.$$

  The adjusted Rand index has an expected value of zero and a maximum value of unity. This means that ARI values around zero indicate a clustering of similar usefulness as that of a random clustering, while index values around unity indicate almost identical grouping.

Correlation Criteria

This final category of external validity indices are based on correlation measurement between the matrices derived by the external partition and the resulting clustering.Let $P(i,j)$ and $C(i,j)$ the similarity matrices that we wish to compare and let p_{ij},c_{ij} their respective elements. We define the following indices:

- Huberts Γ statistic (Γ)

$$\Gamma = N_T^{-1} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (p_{ij} \cdot c_{ij})$$

- Normalized Γ statistic ($\overline{\Gamma}$)

$$\overline{\Gamma} = \frac{N_T^{-1} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (p_{ij} \cdot c_{ij} - \mu_p \cdot \mu_c)}{\sqrt{\left[N_T^{-1} \cdot \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (p_{ij}^2 - \mu_p^2)\right] \left[N_T^{-1} \cdot \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (c_{ij}^2 - \mu_c^2)\right]}},$$

where μ_p,μ_c are the mean vectors of the two matrices. e.g.

$$\mu_p = N_T^{-1} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} P(i,j) \quad , \quad \mu_c = N_T^{-1} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C(i,j).$$

This index can be expressed in terms of the previous section notations as following:

$$\overline{\Gamma} = \frac{N_T \cdot yy - (yy+yn)(yy+ny)}{\sqrt{(yy+yn)(yy+ny)(nn+yn)(nn+ny)}}.$$

Regarding the standard Γ index we have established that large values indicate high similarity between the matrices whereas in the normalized version we seek values around zero.

3.3 Statistical Validation

Throughout this chapter we have discussed about various approaches in order to define sufficient indices to evaluate the clustering result - this part is often called **validity** of the clustering. Another crucial part that completes cluster validation is the part of **reproducibility**. This part answers the question of whether our clustering results can be reproduced in similar independent datasets and is really important in life-science oriented analyses. In the last paragraph of this section we also refer to a technique used in both internal and external indices and attempts to judge how likely is that the observed index value has been achieved by chance.

External indices			
Index	**Abbreviation**	**Range**	**Rule**
Entropy	*Entrp*	$[0,+\infty]$	min
Purity	*Pur*	$(0,1]$	max
Precision	*Prec*	$[0,1]$	max
Recall	*Rec*	$[0,1]$	max
Czekanowski-Dice	*CD*	$[0,1]$	max
Kulczynski	*KU*	$[0,1]$	max
Fowlkes-Mallows	*FM*	$[0,1]$	max
Jaccard	*J*	$[0,1]$	max
Rand	*RI*	$[0,1]$	max
Adjusted Rand	*ARI*	Upper bound: unity	max
Normalized Γ	$(\overline{\Gamma})$	$[-1,1]$	around zero

Table 3.2: Identify good clustering results via external indices

3.3.1 Data Resampling

A very common practice to statistically validate the clustering results is that of *data resampling.*The notion is that we are going to simulate perturbations of the original dataset and assess the stability of the clustering results. The more stable the higher the quality of the clustering schema. Resampling techniques vary usually between *bootstrapping*,*perturbations*, *jacknife* while the choice depends on the data and clustering algorithm used. This approach is also useful to seek the optimal number of clusters in terms of the point where the stability score reveals a local optima.

3.3.2 Prediction Strength

Another approach that in specific areas (such as marketing research) is quite useful is that of assessment of the *predictive strength* of the clustering result. Here, clustering is perceived as a classification problem and we use cross-validation techniques to assess its quality.

In prediction based techniques such as *CLEST* the data set is split into two non-overlapping sets (playing the role of train and test dataset) and subsequently the learning set in clustere producing labels that will be the basis of the classifier. Then the test set is also clustered and the obtained labels are compared to the classifier obtained ones using an external validity index. In more sophisticated approaches we use r-fold cross validation, separating the data set in r distinct parts where each time the r-1 parts play the training set role and the remaining is used as a test set.

3.3.3 Monte Carlo Simulations

The notion for this technique is that a simple value of a validity index may not offer us much in terms of measuring the goodness of fit on the dataset. Thus, we attempt to compute an approximation of the probability density function of the index via Monte Carlo Simulations. In particular a significant amount of synthetic datasets is produced, and for each one of them the validity index of choice is produced. The scatterplot of these indices gives as the approximate pdf which in turn can be used for statistical inference.

4 Comparative Analysis

The scope of this chapter is to perform a comparative analysis of different types of clustering algorithms. The algorithms' performance will be measured in terms of validity indices as they were presented in the previous chapter, while the comparisons will be made in a variety of datasets designed to highlight strengths and weaknesses of the respective algorithms.

4.1 Datasets Generation

The observations of the datasets were obtained as samples from different multivariate normal distributions (MVDs). A variety of covariance matrices were pre-determined in order to achieve the creation of clusters with specified orientation and desirable elongation. Moreover, the mean vectors of the distributions were user-defined in order to achieve (or not) overlapping between the clusters. Finally specific untypical shapes that differ from the elliptic shape of multivariate normal distributions, such as circles and parabolas, are simulated with the use of specific R packages.

In more details, two groups of four different combinations of the aforementioned shapes were simulated. The first group depicted data belonging to five clusters, while the latter depicted ten different clusters.

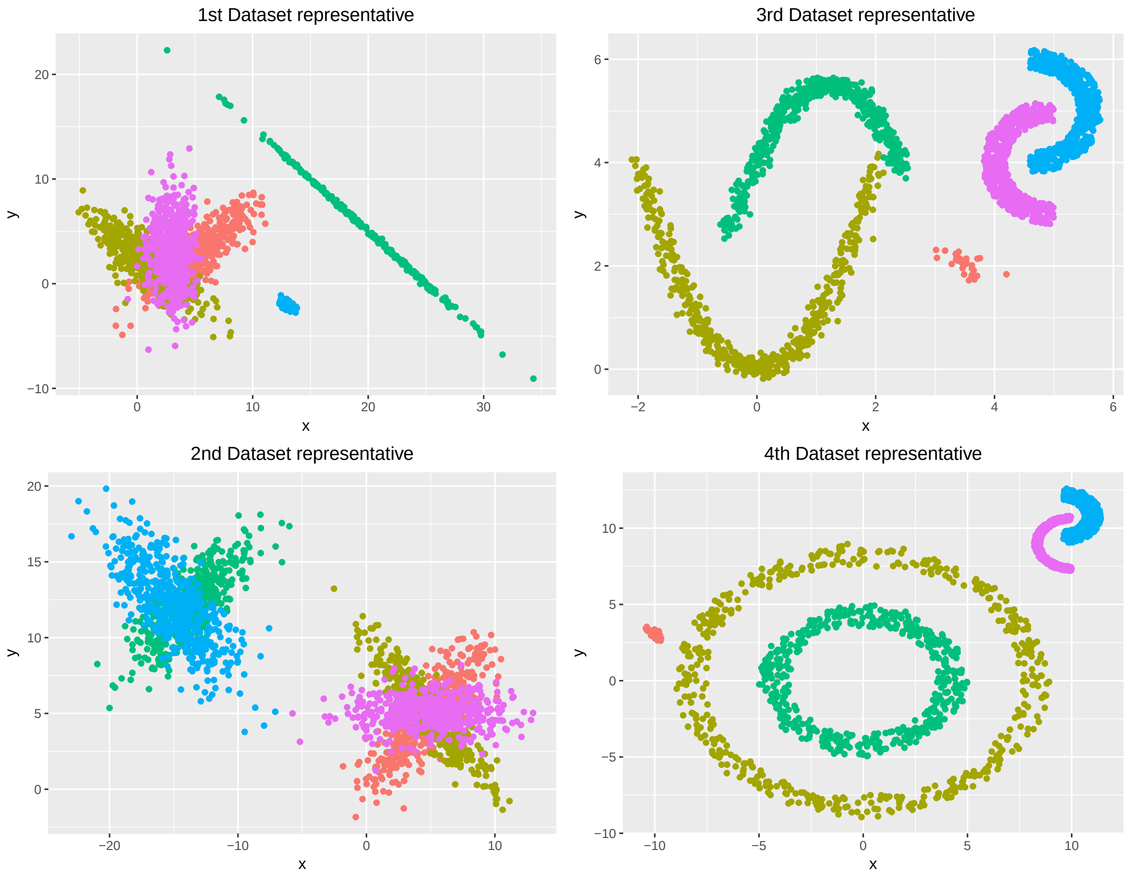


Figure 4.1: 1st Package

The size of each generated dataset varies from 1850 to 3230 data objects, while the density of each cluster within the respective dataset fluctuates as a percentage of the dataset size. For the scope of the comparative analysis it was deemed appropriate to keep the data as two-dimensional, allowing an easy optical evaluation of the results.

A final mention has to be made about the sixth dataset. It is designed not to examine how the different clustering algorithms handle clusters within clusters or non-typical shapes, but what percentage of overlap between different constitutes impossible their correct identification. Specifically, elliptical clusters were generated from MVDs, with initial overlap percentage 5%, increasing up to 25% by step of 5%.

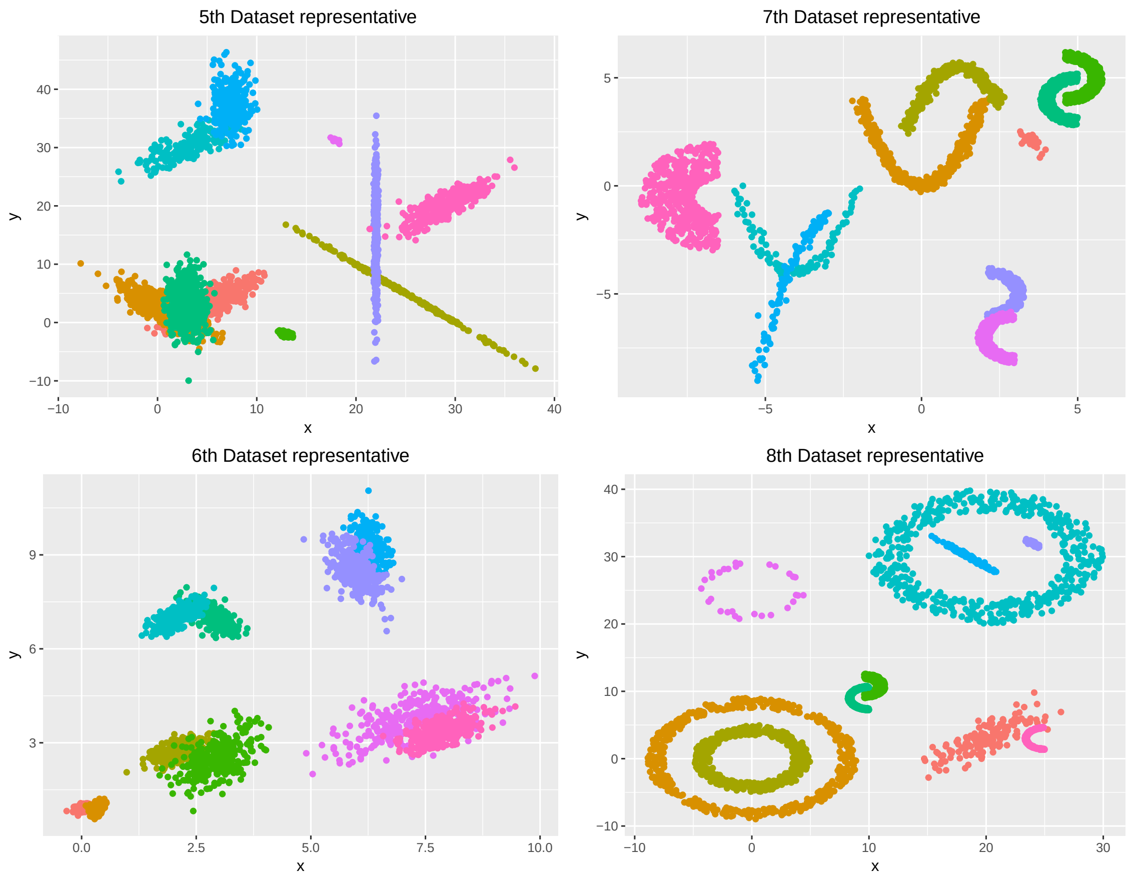


Figure 4.2: 2nd Package

4.2 Parameter Optimization

A very crucial part of the analysis was the appropriate setting of the different clustering algorithms' parameters. Part of the results evaluation was based on whether each algorithm could identify the "correct" number of clusters within the different datasets. This information is usually unknown in real problems and can be replaced with the analyst's intuition, but in our analysis is already pre-determined.

Given the fact that we are going to use the number of identified clusters as a "quality index" makes an absolute need that we cannot bias the initialization of each algorithm leading it indirectly to the correct results. In order to avoid this problem we made use of **relative criteria** technique as defined in the previous chapter. In more details, a reasonable range of the input parameters for each algorithm was selected, while a clustering result was produced for every possible initialization in that range. All the results were then evaluated with the use of the **external indice of Fowlkes-Mallows (FM)** and the best was chosen. The specific details for each algorithm are the following:

4.2.1 K-Means

This particular algorithm is perhaps the most distinct paradigm of why we had to use relative criteria in order not to put bias in our analysis: its sole input parameter is the number of clusters to identify in the dataset.

The range of the parameter was set to be from 3 to 10 by step of one for the first group of datasets (where the real number of clusters is five) and from 3 to 20 with the same step for the second group. In addition, in each run we defined that 20 different random initializations of the cluster centers would be made and then the best would be chosen in terms of lowest within cluster variation. Finally the clustering result with the highest *FM index value* was returned as the optimal.

4.2.2 Agglomerative Hierarchical

In hierarchical clustering methods there are no input parameters to produce the hierarchy other than the distance metric and the linkage criterion (in our analysis both set to Euclidean distance and Ward's criterion). A choice has to be made though on what height the hierarchy tree is going to be cut in order to produce a crisp clustering result.

Instead of setting an exact height the required number of resulting clusters was given as a parameter and the appropriate height was calculated automatically. Specifically, the range of the number of clusters parameter was once again set to be $[3,10]$ for the 1st package of datasets and $[3,20]$ for the 2nd package. The result with the highest *FM index* value was selected as best.

4.2.3 Gaussian Mixture Model

The model-based clustering algorithm implemented in the analysis was that of a mixture (in our case with equal mixture probabilities) of multivariate normal distributions. Most R packages implementing this algorithm use the *Bayesian Information Criterion (BIC)* in order to identify the number of components existing in the dataset,while assuming that there is a one-to-one correspondence between the number of the model components and the number of clusters. Based on the work of *Baudry,Raftery,Celeux,Lo and Gottardo*, and assuming that atypical clusters shapes could be described better by a combination of two or more model components we followed three different meta-analysis steps in order to identify the optimal clustering result. In more details:

Initial Model

For the initialization of the basic model-based clustering we had to select two initial parameters: The *first parameter* was the range of the number of the model components that would be compared based on the BIC criterion. Following the same formalization as on previous algorithms for the 1st group of datasets we used the range of $[1,9]$ while for the second we used $[5,15]$. The ranges were slightly more strict than on the previously described algorithms as BIC selects the number of the mixture components that better approximate the density, and has been proven to slightly overestimate the number of clusters assuming the one-to-one correspondence. The *second parameter* that had to be pre-determined was that of the models allowed to be compared during the EM part of the algorithm. The specification had to be made regarded the geometric characteristics of the identified-to-be components. For the purposes of our analysis we allowed model of varying volume and shape and either equal or varying orientation (*VVE or VVV following the encoding of the* ***mclust*** *R package*).

Optimal clustering result

The first clustering candidate was that of the previously described algorithmic procedure without further modification of the proposed result. We have to mention thought that all the results of the model-based algorithms are *soft clustering* results, assigning probabilities of cluster membership which in turn we used in order to create the required crisp clustering. For the estimation of perhaps even better clustering results we followed three different approaches based though on the same core idea:

Starting from the mixture model proposed by BIC, a sequence of clusterings is created by successively merging two of the components till only one big cluster remains. The criterion based on which the two merged components are chosen is that of minimizing the clustering entropy at each step. It is quite important though to mention that this method does not yield strictly hierarchical clustering results as a data point may not be assigned based on the maximum conditional probability on either of the merged-to-be components but yet be assigned to the resulting merged one.

The optimal clustering result may be chosen as following:

(a) FM index

Assuming that the original number of components is G, following the procedure described above we end up with G different clustering results including the original. Each one of them is evaluated in terms of the FM index and the optimal one is returned.

(b) Piecewise linear fit with one break point (1-BP)

An alternative way of determining an optimal result is to plot the resulting clustering entropy at each step versus the total number of clusters in it and perform a piecewise linear fit in order to identify the elbow in the plot. Taking into account that usually the original solution slightly overestimates the number of clusters in the dataset, after the first couple of component merging the entropy does not drop so drastically. Thus, we identify as the optimal proposed number of clusters, hence the optimal clustering results that is derived by that choice, the spot where the first knee point exists. This is identified easily if we perform a piecewise linear fit with a single break point.

(c) Piecewise linear fit with two break points (2-BP)

In a respective way, in case we need a more strict results that perhaps underestimates the ”real” number of cluster in our dataset we perform a piecewise linear fit on the same plot with two break points and obtain the clustering results derived by the number of clusters of the second knee point.

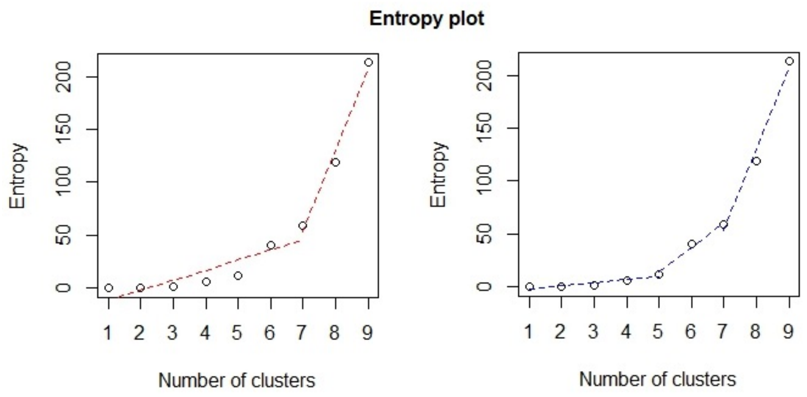


Figure 4.3: (Left): One BP fit returns an optimal number of 7 clusters. (Right): Two BPs fit return an optimal number of 5 clusters, which is our ground truth.

4.2.4 DBSCAN

The parameter optimization of this density-based algorithm has been the most challenging among all the compared ones. The main reason is that we had to deal with two input parameters whose optimal values by trial and error methods varied vastly among the different simulated datasets. The difficulties that had to by surpassed constitute the other side of the same coin that has led to the implementation of more efficient density-cluster algorithms such as HDBSCAN. The optimization procedure implemented can be described as a three-step procedure:

***Eps* optimization**

We recall that the *Eps* parameter denotes the radius around any data point within which a total number of *MinPts* or more have to lie in order to characterize the original data point as a *core point*. In a simple rephrase it is the parameter that,given a *MinPts* value, separates the data space into dense or non-dense areas. It is apparent from the definition that for different datasets and for very distant *MinPts* values the optimal *Eps* value can variate in tremendous ways.

In order to determine the optimal *Eps* value for each value of the second parameter an optical way is proposed in almost every implementation of the algorithm: For each of the data points in the dataset the k-th nearest neighbor distance (where k is each time set as the *MinPts* value) is calculated. Afterwards, a plot of the sorted distances is made, in order to identify a knee point. The distance on which the knee point lies is the optimal *Eps* value, given that specific dataset and *MinPts* parameter. It is obvious that following this technique requires not only user intervention (completely non-automated procedure) but for every different value of the second parameter a completely different optimal value of *Eps* arises.

Our implementation was based on this method although achieved a more automated approach of it. In more details, for a specific *MinPts* parameter the aforementioned distances were calculated,ordered and normalized. Afterwards, their numerical first derivative was calculated (applying the smoothing splines approximation) and the maximum value was stored as an indicator. Continuously a vector of seven different derivative values were set, calculated as a percentage of the maximum value (varying from ten to ninety percent with a step of ten). These derivative values were finally used in order to obtain the original distances, corresponding to the original vector places where the derivative value exceeded each of the seven indicator values for the first time. In that way we had a plausible vector of candidate *Eps* values for each *MinPts* parameter without having to interfere externally to isolate them in each case.

MinPts upper limit

On the previous sub-section we described the method implemented to automatically return a vector of candidate values for the *Eps* parameter, given the *MinPts* parameter. At this very point we could simply set a range for the second parameter and perform a narrowed down grid search for the optimal clustering results. This approach though, as the dataset size increases, would be computationally inefficient.

The idea to bypass this was that we need a small range for the *MinPts* parameter that would fit properly each time to the different datasets. Taking into account that our synthetic datasets do not include any noise, an upper limit of the noise percentage allowed on the clustering result of the DBSCAN could drastically limit the allowed combinations of the algorithms parameters and on the same time adapt to the unique nature of each dataset. Thus, we implemented the DBSCAN clustering into a representative of each dataset, by allowing the *MinPts* parameter to take values into $[0, 100]$ while the *Eps* parameter took the values resulting from the before mentioned technique. Afterwards, we created a plot of the number of points identified as noise versus the value of *MinPts* while the points denoted the different *Eps* values used for each of the second parameters' value. According to each plot we could set meaningful upper limit of the noise percentage that we could allow on the solution and thus set an upper limit on the range of the *MinPts* parameter.

Optimal Clustering Result selection

Following all the above procedure the optimal clustering result was obtained as following:

An upper limit of the allowed noise percentage of the solution was set for each of the eight different datasets. This in turn, leaded to an upper limit of the *MinPts* parameter. For each value of minimum points inside the range set from the above all the nine different values proposed for the *Eps* parameter were tested and the optimal result was obtained by comparison of the *FM index* values it generated.

Dataset	Noise % limit
1st Dataset	10%
2nd Dataset	10%
3rd Dataset	10%
4th Dataset	10%
5th Dataset	30%
6th Dataset	1%
7th Dataset	20%
8th Dataset	30%

Table 4.1: Allowed noise percentage on DBSCAN clustering result for each dataset

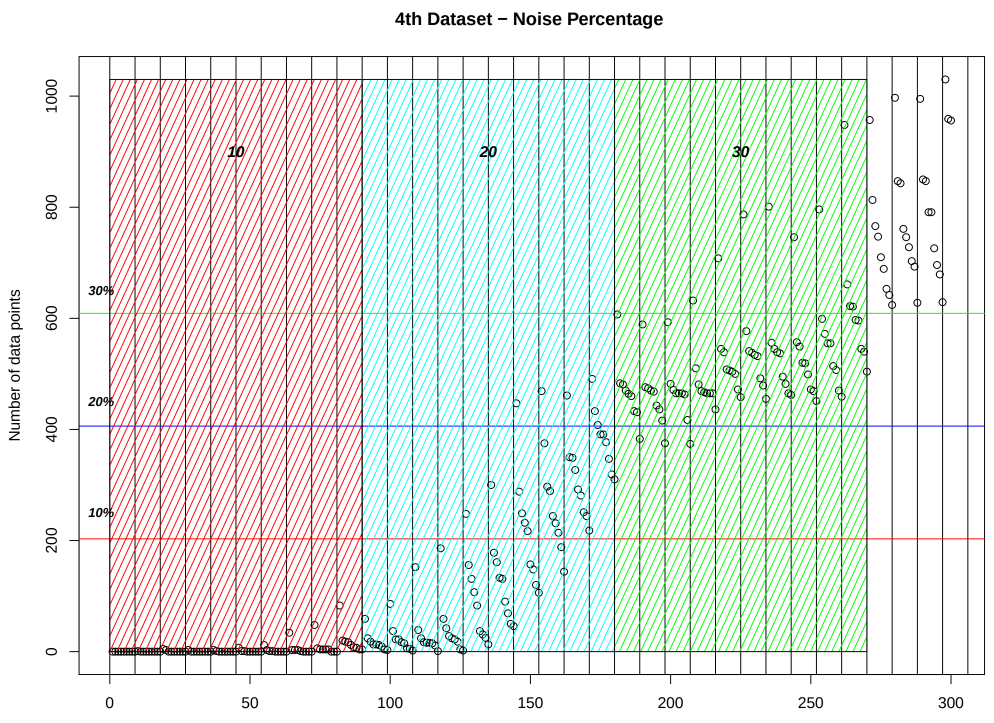


Figure 4.4: A limit of 30% noise in the 4th dataset is surpassed after the value of 21 of the *MinPts* parameter.

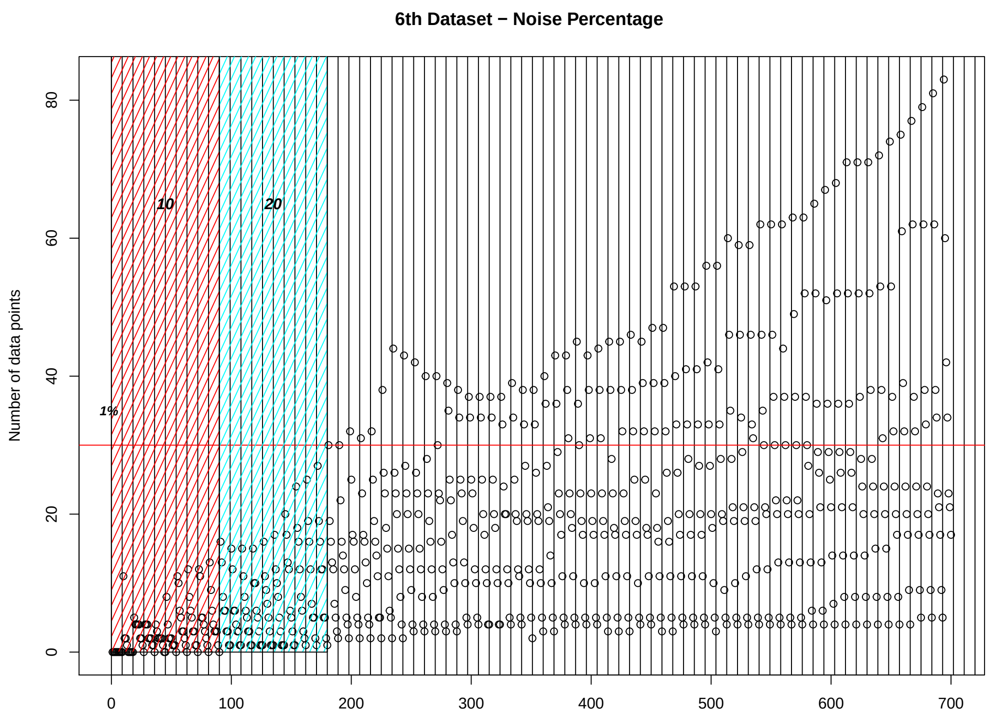


Figure 4.5: A limit of 1% noise in the 6th dataset is met at *MinPts* value of 21 and afterwards.

4.2.5 HDBSCAN

As in most applications of an HDBSCAN clustering algorithm the parameter k used in order to define the core distance of a data point to each k-th nearest neighbour,hence calculate the *mutual reachability distance*, is set to be equal with the *minimum cluster size* parameter.

Across our eight different simulated datasets the actual minimum cluster size was 30 data points. In order not to exclude though, solutions that would be better in terms of the *FM index* by assuming that the smallest cluster is just noise we set as a universal domain interval of the parameter the $[2, 100]$. Finally, considering that not all our datasets have relatively small clusters or that the best solution may have been achieved already in small values of the *MinPts* parameter we set the following **stop criterion**:
After the first 20 iterations (hence after the minimum cluster size of 21) examine the fluctuation of the index value for the last 10 parameter values. In case, the mean difference between consecutive index values is below 0.01 assume that the clustering results have converged to a solution and stop. Among the already performed clusterings return as optimal the one with the highest *FM index value.*

4.3 Results

The evaluation of the clustering results occurred in two phases. On the first phase, we used the *Fowlkes-Mallows* external validity index as a relative criteria in order to obtain the optimal clustering result per algorithm and simulation of each of the eight different type of datasets. On the second phase we assessed the validity of the results based on:

- Percentage of successful identification of the number of clusters

  This is an intuitive validity criterion as it attempts to reveal the ability of each algorithm to correctly identify the real number of clusters in the dataset (which in real datasets in unknown).

- ARI

  To supplement the above ”index” we evaluated the results of each clustering algorithm by measuring the mean value of the Adjusted Rand Index, among the simulations that identified the real number of clusters.

On the following table (Table 4.2) we observe these indices for every dataset of our analysis. It is useful to point out that the parentheses on the GMM results indicate which method of meta-processing yielded the optimal result (the case of original mixture corresponds to *init*) as described and labelled in paragraph subsection 4.2.3.

Algorithms	Index	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7	Dataset 8
K-Means	**ARI**	0.318	-	-	-	-	0.648	-	-
	%	13	0	0	0	0	10	0	0
GMM	**ARI**	**0.496**	**0.355**	0.647	0.480	**0.704**	**0.707**	0.649	0.573
	%	**20 (init)**	**13 (b)**	38 (b)	16 (c)	**17 (init)**	**34 (a)**	14 (b)	14 (b)
Hierarchical	**ARI**	0.315	-	0.665	0.315	0.512	0.681	0.759	-
	%	3	0	22	3	1	20	3	0
DBSCAN	**ARI**	-	-	0.925	0.637	-	-	0.776	-
	%	0	0	17	1	0	0	1	0
HDBSCAN	**ARI**	-	-	**0.963**	**0.999**	-	-	**0.872**	**0.994**
	%	0	0	**81**	**30**	0	0	**2**	**51**

Table 4.2: Clustering results evaluation

As we can recall from Figure 4.1 & Figure 4.2 datasets 1-2 and their respective 5-6 depict mostly elliptical shapes generated by multivariate normal distributions.This is reflected on the clustering algorithms that performed better on that datasets (the bold results per dataset indicate the overall better clustering result) as well. The Gaussian mixture model yielded the better results across the four datasets, while on the datasets 1 and 5 which do not include major cluster overlaps the best results were obtained by the original number of components without further meta-processing. Based though on the results of the *ARI* index which failed to surpass the 0.8 value, we conclude that none of the proposed algorithms can handle cluster overlap efficiently, even in the case that it identifies the correct number of clusters. In particular, the results on dataset 6 indicate that they percentage of overlap between clusters does not play a pivot role on the performance of the various clustering algorithms. Finally it is not strange that the optimal clustering results for this dataset were achieved through the first choice of meta-analysis of the GMM components (*via the relative criteria of FM*), as solutions of reduced entropy implemented in the rest of the approaches, would obviously suggest the merging of the overlapped clusters.

On the other hand in datasets 3-4,7-8 where more utypically shaped clusters exist, the density based algorithms seem to have the most qualitative results. In particular we can observe that indeed HDBSCAN outperforms DBSCAN as on these datasets we have a variety of disproportionally sized clusters and a variety of densities which DBSCAN cannot handle efficiently. This superiority becomes more clear by observing the results of Table 4.3, where for each dataset and algorithm we present the identified number of clusters found on the majority of the simulations, along with the respective percentage (e.g. K-means algorithm for the second dataset identified in all simulations 3 clusters). The absent values for the DBSCAN algorithm indicate the failure of the algorithm to converge to a single solution.

A final conclusion that can be drawn from Table 4.3 is the one that refers on the meta-processing of the GMM clustering results. We can observe that the original solution implemented in most packages slightly or greatly overestimates the number of clusters in the dataset. Thus, they hypothesis made that these better depict the number of model components and have to be further analysed in order to extract the number of clusters is validated from our results. Moreover, it is apparent that datasets that include non-typical cluster shapes only may not need further analysis and can be slightly improved by seeking a solution with the external relative criteria of *Fowlkes-Mallows*. Finally depending on how strict our clustering solution is required to be, we can make a choice between the final two approaches based on the reduction of the clustering solution's entropy.

Algorithms	Index	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7	Dataset 8
K-Means	**nc**	**4**	**3**	**3**	**4**	**5**	**13**	**7**	**4**
	%	83	100	66	50	26	25	49	100
GMM (init)	**nc**	**6**	**7**	**9**	**9**	**11**	**11**	**15**	**15**
	%	57	30	96	98	33	34	86	99
GMM (a)	**nc**	**3**	**2**	**5**	**2**	**11**	**10**	**7**	**5**
	%	99	100	30	97	23	34	31	38
GMM (b)	**nc**	**3**	**4**	**5**	**7**	**8**	**6**	**11**	**8**
	%	88	38	38	28	58	60	36	21
GMM (c)	**nc**	**3**	**2**	**3**	**4**	**8**	**5**	**6**	**5**
	%	98	59	53	38	48	77	45	35
Hierarchical	**nc**	**3**	**3**	**3**	**3**	**7**	**5**	**6**	**4**
	%	62	100	61	62	28	62	40	100
DBSCAN	**nc**	-	-	**4**	**4**	-	**5**	**6**	-
	%	-	-	69	89	-	99	65	-
HDBSCAN	**nc**	**3**	**2**	**5**	**4**	**5**	**5**	**8**	**10**
	%	99	100	81	69	58	100	59	51

Table 4.3: Dominating solution per dataset

5 Product Data Application

5.1 Dataset Description

The data used in the current chapter are a property of the company **Information Resources Hellas (IRi Greece)** and have been made available for the scope of our co-supervised project. For that reason, all of the results have been anonymized and only the necessary information for the implemented methodology are presented.

The dataset consists of two hundred and six (206) unique product codes, along with a set of nine (9) features for each one of them. The different features depict natural product characteristics and manufacturing details. All of the features are handled as factor variables, while two amongst them are treated specifically as ordinal. Finally, for each one of the product codes we have access to a 40-dimensional representation of them (acquired by a pre-determined procedure foreign to the topic of the thesis), which will play the role of a measure of similarity between them. In Table 5.1 we present the number of levels of each factor variable.

Feature	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9
Number of Levels	4	27	10	13	2	11	13	10	5

Table 5.1: Number of factor levels

5.2 Procedure Description

5.2.1 Goal of application

The idea behind the application is to ”borrow” the notion of the *clustering quality index* and apply it on a dataset of retail products in the following manner:

Each of the initial product features is considered to partition our data space by a usage of its own factor levels as cluster labels. Thus, we can depict the quality of the respective partition with a use of clustering quality index and seek the feature that achieves the optimum result. The realization of this feature inevitably leads to sub-groups of the original dataset, where we can apply the same technique with the exception of excluding the already optimum ”parent” feature. In that way, a form of a decision tree arises where each sub-group is more and more qualitative.

The application of this technique can be made on raw product data but can also be used in already formed clusters by a clustering algorithm. This option, in particular, can be very useful in cases where the amount of available data is tremendous and there is need to perform an initial clustering in order to obtain lower volumes of data that can be used for business decisions.

5.2.2 Data objects’ distances

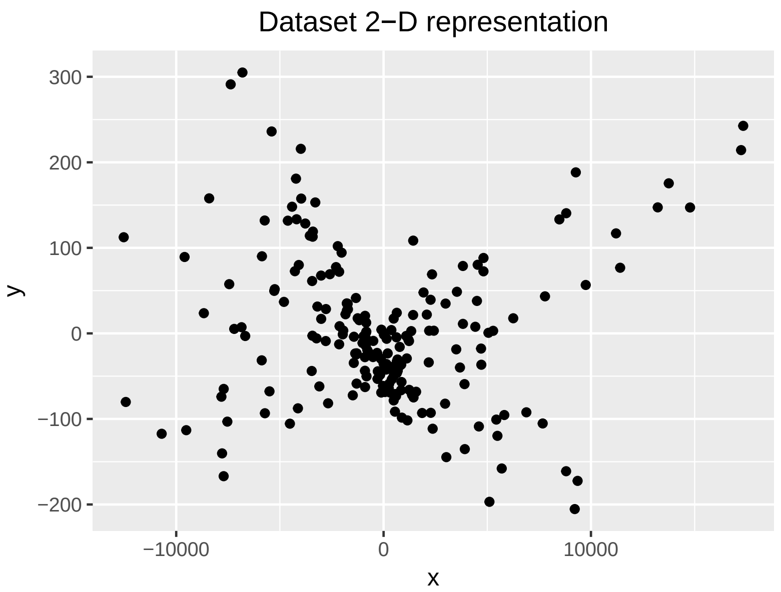


Figure 5.1: Dataset 2-D plot

Figure 5.1 illustrates our data space with the use of the first two (out of the 40 offered) coordinates. It is apparent from the plot that the coordinates require a scaling transformation. This can be achieved by selecting as a dissimilarity measure between the data objects the **Mahalanobis distance.** In particular the distance between any two data objects with respective distance vectors **x**,**y** is calculated as:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T S^{-1}(\mathbf{x}-\mathbf{y})},$$

where S^{-1} is the covariance matrix of the coordinates dataset.

Despite, that the *Mahalanobis distance* was the selected normalization procedure during our application, it is worth noticing that it yields equivalent results with the *Euclidean distance* in case of previously applying to the coordinate vectors a *min-max normalization.*

5.2.3 Validity index selection

In most problems of real data clustering the general truth about the underlying structure of the data space remains unknown. In our case study there was no intention to predetermine some labels of the given features that business knowledge has deemed similar and thus could be merged. This narrows down the choices for the index only to *internal* ones as defined on subsection 3.2.2.

Most *internal validity indices* measure the clustering quality in terms of compactness and separation of clusters. As we have observed on Figure 5.1 there is a significant overlap between data objects, which after examination of the initial partitions derived from the different features cannot be surpassed even in the case of minimal number of cluster labels (see Figure 5.2).

This major overlapping can cause significant numerical stability issues on most of the indices presented on subsection 3.2.2. Indeed our initial selection of the *Silhouette score* failed as from the very early stage of the procedure, computation of the index returned ”NaN” values. This derives from its definition as for the calculation of the b_i coefficient, distances with elements of the nearest cluster have to be calculated which in our case would be of the same magnitude with the intra-cluster distances even at the level of the fourth decimal point.

In order to bypass this difficulty we decided to use an internal indices of a different category that would be much less susceptible to numerical instabilities; the **tau index.** This internal index as defined and described in page 22, is more of a correlation measure and attempts to assess how well the proposed clustering depicts the data objects dissimilarities. It takes values in the interval $[-1,1]$, while values above 0.75 are considered indicators of excellent clustering quality.

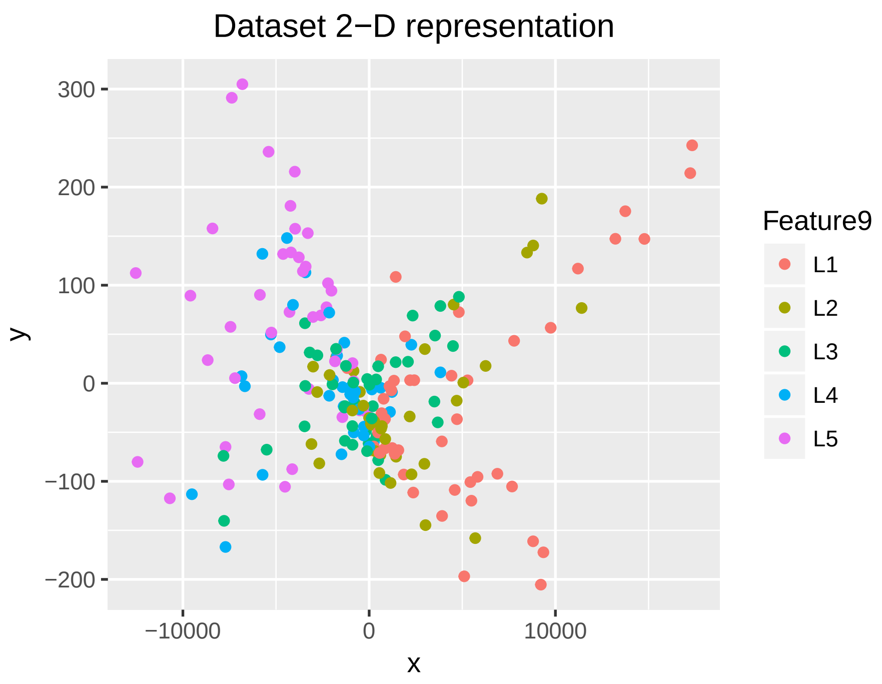


Figure 5.2: Dataset 2-D plot (coloured with respect to the factor labels in the 10th feature)

5.2.4 Technique implementation

Returning to the goal of our application, we want to establish for each of the features the best partition of our dataset. In order to do that, we have to perform a grid search that would reveal the possible label merges within each feature that increase the value of the *tau index*. This can be really demanding computationally as the possible different combinations of the factor levels are given by:

$$\sum_{i=2}^{n} \binom{n}{k} \quad \forall n > 2,$$

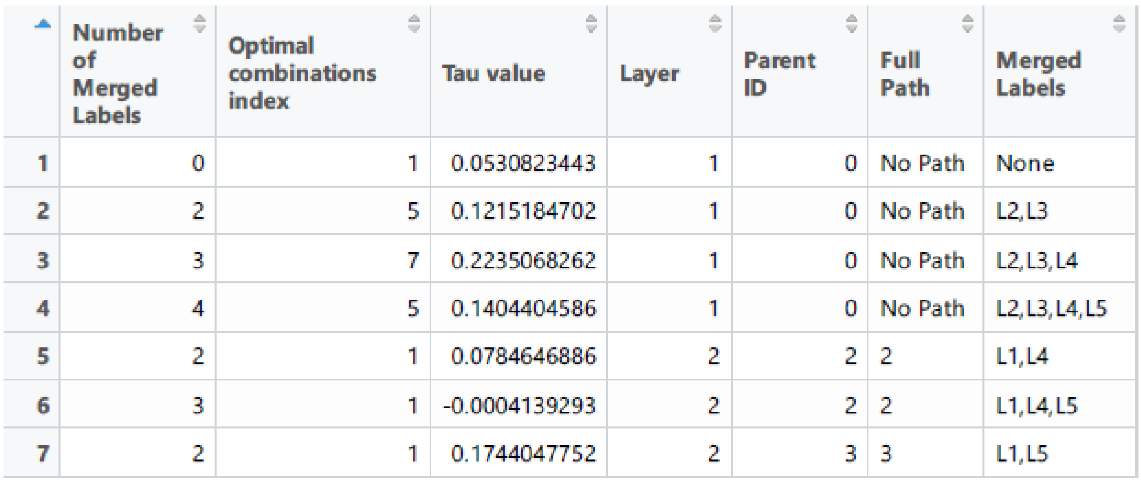
where n is the number of levels. This number of possible combinations would yield $n-1$ proposed merges including the one that does not change the original labels but excluding the one that would result in one big cluster. (e.g. a factor with 5 levels would return one result for the best quart of labels,one result for the best triplet, one for the best couple and one without any merges, after computing and comparing in terms of *tau index* 26 different options)

Up until this point though we evaluate only first layer merges. The optimum index value though could be a result of successive possible label merges, as after the first one there are still original labels left that can still be further merged. In order to avoid complexity and not re-evaluate solutions already visited we set some new conditions after the first suggested label merge:

- We do not allow a solution where the remaining labels remain at the initial situation (in contrast to the first step) & we allow a solution that would merge the remaining labels to one.
- There have to be two or more remaining labels in order to continue the path.
- The already formed merges in the path are not allowed to participate in the possible following merging procedure.

Following that procedure we conclude that a feature with n labels can explore solutions up to the *integer division of n with 2* ($n\%/\%2$ in R notation) layer. Revisiting the above sum of possible combinations, and by defining as c_n and s_n the sum of combinations and proposed solutions respectively of the search within a feature with n levels we have:

$$\begin{cases} c_n = 2 \cdot c_{n-1} + s_n, n = 3, \ldots, \quad c_2 = 1 \\ s_n = s_{n-1} + s_{n-2} + 1, n = 4, \ldots, \quad s_2 = 1, s_3 = 2 \end{cases} .$$



	Number of Merged Labels	Optimal combinations index	Tau value	Layer	Parent ID	Full Path	Merged Labels
1	0	1	0.0530823443	1	0	No Path	None
2	2	5	0.1215184702	1	0	No Path	L2,L3
3	3	7	0.2235068262	1	0	No Path	L2,L3,L4
4	4	5	0.1404404586	1	0	No Path	L2,L3,L4,L5
5	2	1	0.0784646886	2	2	2	L1,L4
6	3	1	-0.0004139293	2	2	2	L1,L4,L5
7	2	1	0.1744047752	2	3	3	L1,L5

Figure 5.3: Result example for feature 9 - The optimal result occurs by merging L2,L3,L4

After we have successfully obtained all the proposed solution for a feature we keep the best in terms of the value index and compare it with the respective ones of the remaining features. By selecting the global maximum, we can reconstruct the path of merges that led to it at that futures and impose them on the original dataset. In turn the new labelling of the feature, subsets the original dataset in subsets which contain only one of the remaining labels (original or newly formed). Each of that subsets is then used as an input to the described procedure with the exception that the feature(s) already used for its formation up to the root of the forming tree, are not explored again.

The building of that tree either stops after no further improvement can be achieved or when one of our stop conditions apply. The stop conditions can vary from the number of leaves, the depth of the tree, the minimum subset size formed to a more sophisticated business rule.

5.3 Results

5.3.1 General settings - Stop conditions

Before the implementation of the procedure we set some limitations, considering our initial dataset its characteristics. In more details, and due to the limited number of products in the dataset, newly formed cluster with 10 or less products were considered as final. Moreover, as we observe on Table 5.1 Feature 2 includes 27 different levels. A grid search for an optimal cluster result sourced by this feature would require the comparison of *267.089.268* combinations. Due to the enormous computational effort required for these combinations and making a hypothesis that a feature so fragmented would be highly unlikely to provide an optimum result, we set an upper limit on the number of levels that a feature can have in order to be included in the search - *15 levels*.

5.3.2 2-Dimensional Coordinates

The first implementation took into account only the first two dimensions of the coordinate data in order to create the necessary for the calculation of the *tau index* distances. Figure 5.4 illustrates the feature selected at each level in order to accomplish a better partition of the dataset. As we can observe the maximum path length is seven, with *Feature 1* and *Feature 5* completely absent from the decisions. For the latter feature we could have expected not to play a pivot role, as it consists only of two levels and in the case they partitioned the data space adequately it would have resulted on each selection from the very beginning of the process.

The **total number of discovered clusters**, found after the implementation of these 16 breaks, **are 27**. This happens because almost in every break point clusters of size less than 10 objects are created.

Break Point	**Feature Selected**	**Tree Level**	**Original Tau Value**	**Optimum Tau Value**
1	7	1	0,0607	0,2181
2	6	2	0,1736	0,1974
3	8	3	0,0417	0,1321
4	2	4	-0,0373	0,1594
5	9	3	0,0447	0,1697
6	8	4	0,0193	0,1000
7	2	5	-0,1390	0,1827
8	4	4	-0,0538	0,2056
9	2	5	0,1391	0,2914
10	3	6	-0,0533	0,2837
11	3	4	0,0176	0,2212
12	8	5	0,2907	0,3520
13	8	5	0,0132	0,2359
14	4	6	-0,0186	0,1397
15	2	7	0,0630	0,1787
16	2	7	0,1719	0,2595

Table 5.2: Index values before and after optimization (2-D)

Table 5.2 presents the improvement of the index's value deemed to be optimal across available features at each point. There seems to be no direct relationship between the tree level and the amount of improvement achieved, while the best overall index value obtained is around 0.35 which is a quite medium level result.

5.3.3 40-Dimensional Coordinates

Moving forward to the full dimensional data we expect to have increasing performance results as we are equipped with a better dissimilarity measure. Figure 5.5 illustrates the respective feature selection tree, which once again has a maximum depth of 7 and interestingly enough does not include *Feature 1* as well.

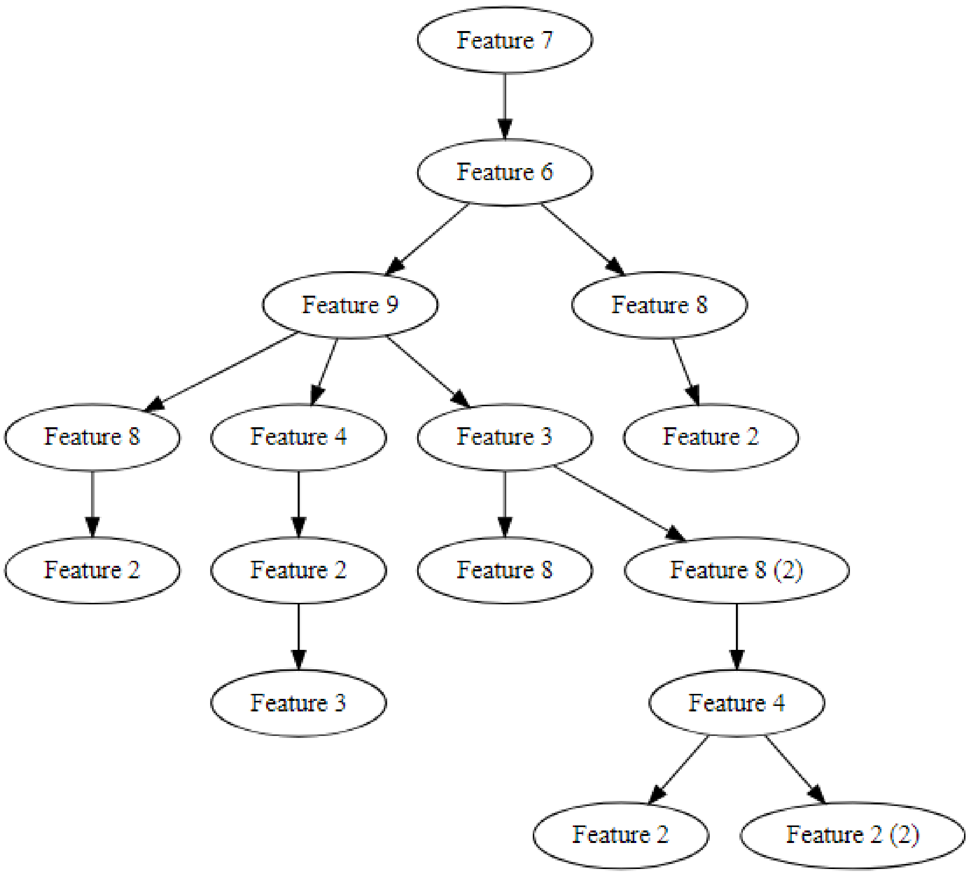


Figure 5.4: 2D Tree of optimal features at each break point

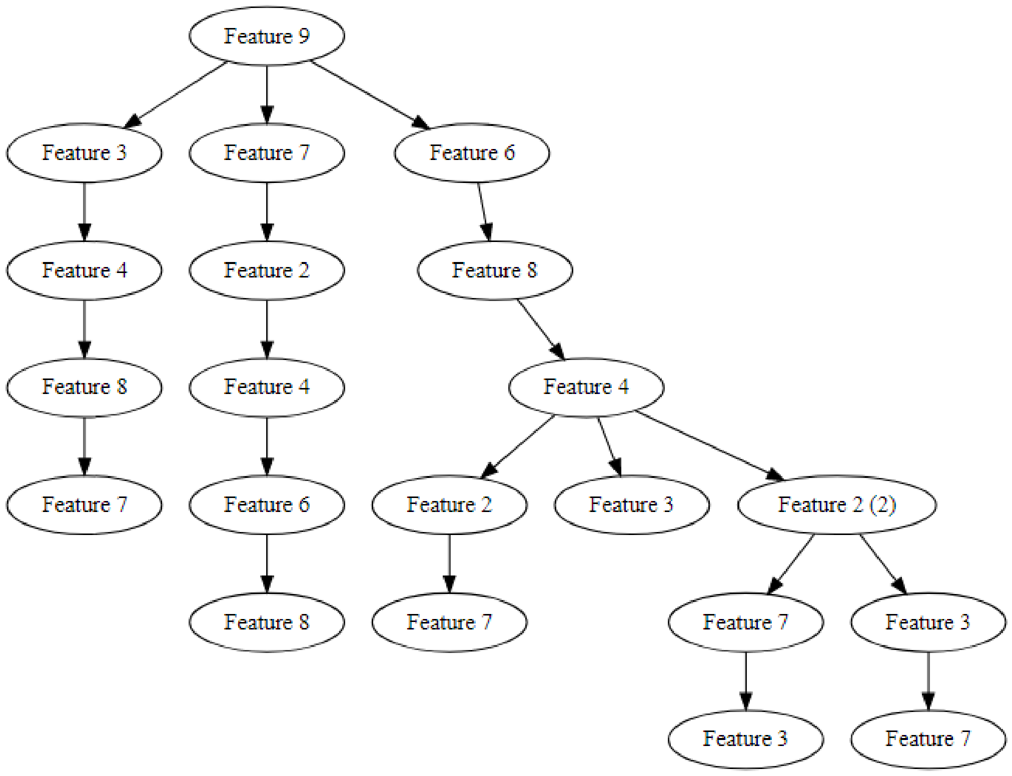


Figure 5.5: 40D Tree of optimal features at each break point

Feature	**2-D**	**40-D**
1	5	7
3	7	5
4	6	6
5	8	8
6	2	4
7	1	3
8	4	2
9	3	1

Table 5.3: Rank of Features in the initial choice step

This can be explained through Table 5.3, where we can see that in the first comparison of the best indices for every available feature (expect *Feature 2* which violates the rule of maximum 15 levels at the point), it ranks really low in both cases. This fact in combination with the low number of levels which after a few partitions will probably be completely separated makes it a highly unlikely candidate for any step.

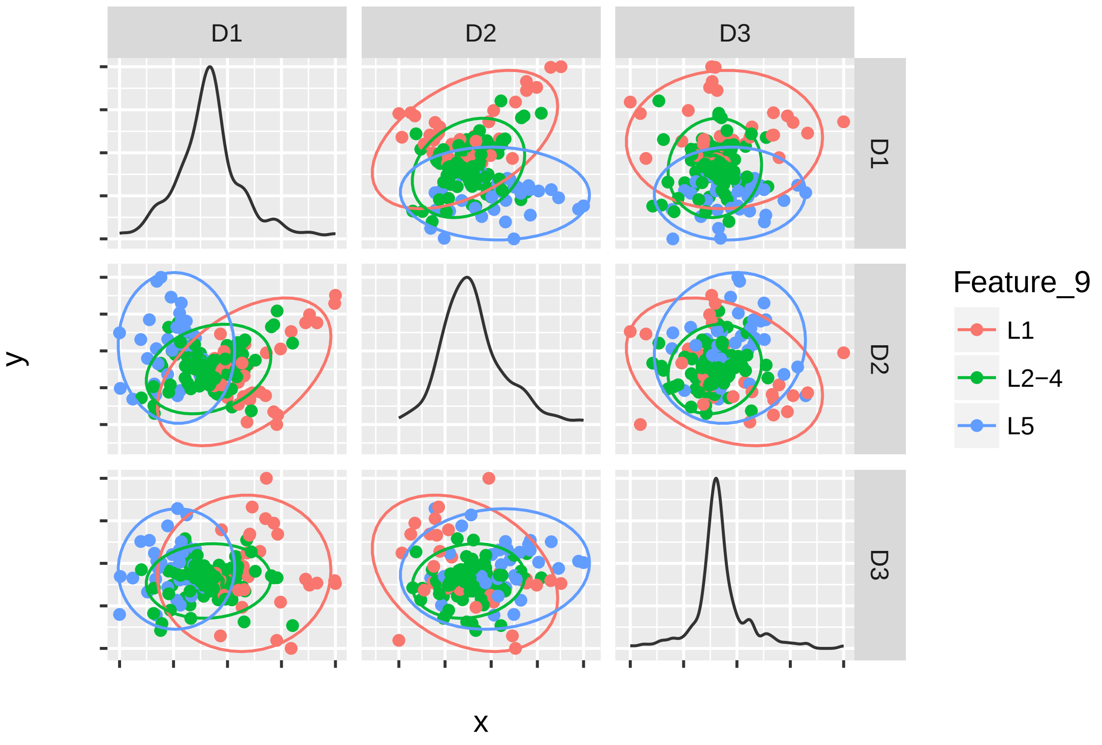


Figure 5.6: Proposed clusters at first step (3 out of 40 dimensions)

Break Point	**Feature Selected**	**Tree Level**	**Original Tau Value**	**Optimum Tau Value**
1	9	1	0,05308	0,22351
2	6	2	0,13115	0,21386
3	7	2	0,12060	0,21018
4	3	2	0,00663	0,04593
5	2	3	0,11250	0,36705
6	8	3	-0,03416	0,09031
7	4	3	0,02776	0,07635
8	4	4	0,04505	0,42656
9	8	4	0,01003	0,10660
10	4	4	-0,01098	0,09586
11	6	5	0,43785	0,52388
12	3	5	0,12437	0,21974
13	2	5	0,10242	0,21836
14	2	5	0,03806	0,13152
15	7	5	-0,17194	-0,06875
16	7	6	0,23031	0,40998
17	8	6	0,23031	0,40998
18	3	6	0,00227	0,36882
19	7	6	0,00874	0,19508
20	7	7	0,07259	0,20830
21	3	7	-0,01601	0,16327

Table 5.4: Index values before and after optimization (40-D)

Figure 5.6 illustrates the overlaps between the three proposed clusters at the end of the first step. As we can observe on the plots of the first three dimensions the overlap of the labels that were originally present on the dataset in minimal, which adds the quality to that solution.

A **total number of 34 clusters** have been identified after the completion of the procedure. As Table 5.4 presents we have 5 more break points than the 2-D solution but we have much more clear solutions. The values of the optimal *tau index* at each step are quite better than the previous solution's, while the maximum index values reaches the level of 0,5.

5.3.4 Conclusion - Further Improvements

The comparison between the partial and full coordinate solution indicates that we obtain much better results when using the full information. It gives us though a far more segmented dataset that perhaps escapes from the scope of the business purpose. In that manner we could impose a far more set of strict rules on the implementation, such as the restriction of minimum size of produced clusters which in turn could lead to the bypass the selection of the optimal feature at one step and move to the next best one. Finally, the part of the grid search along all the possible label combinations could be programmed for parallel processing, as each machine could perform one of the initial steps and then move forward the path without affecting the rest of the procedure.

List of Tables

3.1 Identify good clustering results via internal indices 25
3.2 Identify good clustering results via external indices 29

4.1 Allowed noise percentage on DBSCAN clustering result for each dataset . 38
4.2 Clustering results evaluation 41
4.3 Dominating solution per dataset 43

5.1 Number of factor levels 44
5.2 Index values before and after optimization (2-D) 50
5.3 Rank of Features in the initial choice step 52
5.4 Index values before and after optimization (40-D) 53

List of Figures

4.1 1st Package 32
4.2 2nd Package 33
4.3 (Left): One BP fit returns an optimal number of 7 clusters. (Right): Two BPs fit return an optimal number of 5 clusters, which is our ground truth. . 36
4.4 A limit of 30% noise in the 4th dataset is surpassed after the value of 21 of the *MinPts* parameter. 39
4.5 A limit of 1% noise in the 6th dataset is met at *MinPts* value of 21 and afterwards. 39

5.1 Dataset 2-D plot 45
5.2 Dataset 2-D plot (coloured with respect to the factor labels in the 10th feature) 47
5.3 Result example for feature 9 - The optimal result occurs by merging L2,L3,L4 48
5.4 2D Tree of optimal features at each break point 51
5.5 40D Tree of optimal features at each break point 51
5.6 Proposed clusters at first step (3 out of 40 dimensions) 52

Bibliography

[1] Jean-Patrick Baudry et al. “Combining mixture components for clustering”. In: *Journal of computational and graphical statistics* 19.2 (2010), pp. 332–353.

[2] Guy Brock et al. “clValid, an R package for cluster validation”. In: *Journal of Statistical Software (Brock et al., March 2008)* (2011).

[3] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. “Density-based clustering based on hierarchical density estimates”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2013, pp. 160–172.

[4] Malika Charrad et al. “Package NbClust”. In: *J. Stat. Soft* 61 (2014), pp. 1–36.

[5] Ivan Gesteira Costa Filho. “Mixture models for the analysis of gene expression: integration of multiple experiments and cluster validation”. PhD thesis. PhD thesis, Freie Universität Berlin, 2008. Chap. 2.

[6] Bernard Desgraupes. “Clustering indices”. In: *University of Paris Ouest-Lab ModalX* 1 (2013), p. 34.

[7] Justin Eldridge, Mikhail Belkin, and Yusu Wang. “Beyond hartigan consistency: Merge distortion metric for hierarchical clustering”. In: *Conference on Learning Theory*. 2015, pp. 588–606.

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

[9] Chris Fraley, Adrian E Raftery, et al. “Model-based methods of classification: using the mclust software in chemometrics”. In: *Journal of Statistical Software* 18.6 (2007), pp. 1–13.

[10] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. “On clustering validation techniques”. In: *Journal of intelligent information systems* 17.2 (2001), pp. 107–145.

[11] Christian Hennig. “Methods for merging Gaussian mixture components”. In: *Advances in data analysis and classification* 4.1 (2010), pp. 3–34.

[12] D Karlis. “Multivariate statistical analysis”. In: *Athens: Stamoulis Publications (in Greek)* (2005).

[13] Vipin Kumar. *An Introduction to Cluster Analysis for Data Mining, 2000.*

[14] Csaba Legany, Sandor Juhasz, and Attila Babos. "Comparison of cluster validation methods." In: *WSEAS Transactions on Information Science and Applications* 3.3 (2006), pp. 502–509.

[15] Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra. "MixSim: An R package for simulating data to study performance of clustering algorithms". In: *Journal of Statistical Software* 51.12 (2012), p. 1.

[16] Fionn Murtagh and Pierre Legendre. "Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm". In: *arXiv preprint arXiv:1111.6285* (2011).

[17] Jian Pei et al. *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings*. Vol. 7819. Springer, 2013, pp. 160–172.

[18] Christine Preisach et al. *Data analysis, machine learning and applications*. Springer, 2008.

[19] Jorge M Santos and Mark Embrechts. "On the use of the adjusted rand index as a metric for evaluating supervised classification". In: *International Conference on Artificial Neural Networks*. Springer. 2009, pp. 175–184.

[20] NCSS Statistical Software. *Hierarchical Clustering /Dendrograms*.

[21] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining. 1st*. https://www-users.cs.umn.edu/ kumar/dmbook/ch8.pdf. 2005.

[22] Swiss Federal Institute of Technology. *Cluster Analysis*. Applied Multivariate Statistics Notes.

[23] Silke Wagner and Dorothea Wagner. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[24] Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012. Chap. 5.

[25] Ka Yee Yeung and Walter L Ruzzo. "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data". In: *Bioinformatics* 17.9 (2001), pp. 763–774.

[26] Qinpei Zhao. "Cluster validity in clustering methods". In: *Publications of the University of Eastern Finland* (2012).

[27] Ying Zhao and George Karypis. *Criterion functions for document clustering: Experiments and analysis*. Tech. rep. Technical report, 2001.