MASTER IN COMPUTER SCIENCE

Clustering with Decision Trees: Divisive and Agglomerative Approach

Castin, Lauriane

Award date:
2017

Awarding institution:
University of Namur

Université de Namur
Faculté d'informatique
Année académique 2016–2017

Clustering with Decision Trees: Divisive and Agglomerative Approach

Lauriane Castin

Supervisor: _____ (Signed for Release Approval - Study Rules art. 40)
Benoit Frenay

A thesis submitted in the partial fulfillment of the requirements
for the degree of Master of Computer Science at the Université of Namur

Abstract

Decision trees are tools used in Machine Learning mainly to perform classification tasks. Samples are submitted to a test in each node of the tree and guided through the tree based on the result. Decision trees can also be used to perform clustering, with a few adjustments: On one hand, new split criteria must be discovered to benefit from the samples features instead of the labels to construct the tree, and on the other hand the nodes referring to the same clusters must be merged.

The goal of this thesis is to propose solutions on those two aspects. Unfortunately, when the clustering with decision trees is compared to the existing clustering techniques, the results are poor, with higher misclassification rates on the datasets representing handwritten digits.

keywords: Machine Learning, Decision Tree, Clustering, Split Criterion, Agglomeration

Les arbres de décision sont des outils utilisés en Apprentissage Automatique pour réaliser des tâches de classification. Les échantillons sont soumis à un test dans chacun des noeuds et sont guidés au travers de l'arbre sur base du résultat. Les arbres de décision peuvent aussi être utilisés pour faire du clustering, à condition d'y apporter quelques adaptations: d'un côté, de nouveaux critères de division doivent être découverts pour profiter des caractéristiques des échantillons plutôt que de leurs classes, et d'un autre côté les noeuds faisant référence aux mêmes clusters doivent être fusionnés.

Le but de ce mémoire est de proposer des solutions à ces deux aspects. Malheureusement, quand le clustering avec les arbres de décision est comparé aux technique de clustering existantes, les résultats sont décevants, avec des hauts taux de mauvaise classification sur des ensembles de données représentant des chiffres écrits à la main.

mots-clefs: Apprentissage Automatique, Arbre de Décision, Clustering, Critère de Division, Agglomération

Acknowledgements

Because innovation does not come without collaboration...

I would like to thank my supervisor, Mr Benoit Frenay, for his valuable advices and inspiring discussions during our regular meetings. Thanks to his great knowledge on the topic, he guided my research by asking good questions, challenging intermediate results and proposing new ideas. I am also grateful for the involvment he has shown by reading draft versions of this document and by following my work during the whole year.

My acknowledgements are also addressed to Mrs Anthony Cleve and Wim Vanhoof for the time and expertise put into the reading and evaluation of this thesis. Hopefully, the content of it will live up to their expectations.

Eventually, I would like to thank the numerous friends of mine that carefully reviewed this document, both on the technical and literary aspects. Their remarks were relevant and I certainly owe them the quality of this document: Pierre-Alexandre Beaufort, Gilles Bodart, Henri Bourmorck, Benoit Daene, Natalia Dementieva, Jean-Mathieu Duchêne, Juliana Gonzalez, Geoffroy Herbin, Pierre Joskin, Ariane Liteanu, Basile Maldague, Bryan Mayné and Vincent Tonus.

CONTENTS

CHAPTER

1

INTRODUCTION

The only certainty I had before I clicked on the button to display the list of all master thesis topics proposed by the professors was that I wanted to do some machine learning. Even though I did not know much on the topic, the few courses I took on it caught my attention and I wanted to learn more.

I saw a topic on clustering with decision trees that seemed more or less related to what I was looking for... Clustering? That I know! I know clustering is a technique that aims at finding groups, called clusters, among large amount of data. I can even represent clusters as several clouds of points into a graph. And I have heard of k-means, the famous clustering method, before, that's most of it.

But decision trees? No idea! I quickly discovered what this method was made of: Organized as a hierarchy, each node is responsible for a test and based on the test result, the sample (that's how I call a piece of data) is guided to the left or right branch until it reaches a leaf at the bottom of the tree. At this stage, the sample is assigned the label of the leaf node, i.e. the label of that cluster. And as easy as it can be, that's how clustering works with decision trees. Interesting, isn't it?

Currently, in the literature, the use of decision trees to perform clustering is quite limited. Most papers compute the test results from the variance of the data or derived methods. The challenge Mr Frenay gave me was to find other split criteria, other tests to submit the samples to when they cross nodes in the trees. The challenge was, in other words, to find the divisive approach.

Yet, there was also another obstacle. When the samples reach the bottom of the tree, there may be more leaf nodes than the actual number of clusters, so there is a need for merging the nodes together if they refer to the same cluster. That was the second challenge Mr Frenay gave me: to find the agglomerative approach.

Enough talking, let's jump to the heart of this exciting topic!

This thesis starts with a chapter presenting the key concepts of clustering and decision trees, as well as a literature review on the use of decision trees to perform a clustering task. The second chapter introduces my own contributions to this topic: How to split the nodes and then merge the leaves of the tree. The methods chapter shows the different steps of the algorithm, giving precision on the technical approach for each one of them. Lastly, I present the results of this research and a discussion on the new clustering technique introduced in this thesis as compared to existing ones.

CHAPTER

2

STATE OF THE ART

This chapter sets out the key concepts of clustering and introduces decision trees, in a way that the reader will gain a better sense of the underlying theory and algorithms. The beginning of the chapter helps to understand the purpose of clustering in general. Then, decision trees will be presented: What are they and how are they built? The last part of the chapter combines the two previous ones by reviewing how clustering is performed with decision trees in the literature.

2.1 What is Clustering?

After a short definition of clustering and its purposes, this section describes the famous existing clustering algorithms. Those algorithms are not based on decision trees but can be used as a reference to validate the algorithm developed for this thesis.

2.1.1 Definition

"Cluster analysis aims at identifying groups of similar objects and, therefore helps to discover distribution of patterns and interesting correlations in large datasets." [7]

This definition means that the dataset is divided into several groups, called clusters, in such a way that clusters consist of samples that are similar to each other, while those samples are dissimilar from samples belonging to other clusters [20]. A cluster can therefore be seen as a group of samples resulting from the partition or division of the data.

A small example can help to illustrate, considering an algorithm that must group thousands of images of handwritten digits by the figure they represent: One group with all the

zeros, one group with all the ones, etc. If the target, i.e. the actual figure represented on the image, is known and is used by the algorithm, the problem is a classification problem, a supervised learning problem. On the contrary, if the target is unknown, the algorithm can only take advantage of the characteristics of the image, the pixels intensities, to form the groups. This last problem is a clustering problem, an unsupervised learning problem. Each image is called a *sample* and in this case the pixels intensities are the *features*.

Formally speaking, unlike classification, clustering is part of unsupervised learning algorithms: Each sample is not associated to a pre-defined label, a target. No example can show the desirable relation to be detected among the data points [1], the clustering algorithms can only consider the features of each sample to find a pattern. However, in both clustering and classification tasks, the output is a discrete labelling.

Clustering algorithms play a role in statistics, speech and character recognition, image segmentation and computer vision, data compression in image processing, data mining, pattern recognition and lots of other fields. Clustering applications include spatial data analysis, DNA sequencing in computational biology or web applications, among many others [1].

2.1.2 Famous Clustering Techniques

Apart from the decision trees that will be largely discussed in Section 2.2, this section presents two categories of clustering techniques: k-means and hierarchical clustering.

k-means k-means algorithm is the most famous clustering algorithm, it was designed during the eighties. The number of clusters to find among the dataset is known beforehand. As many centroids as the number of clusters are randomly located in the dataspace. Each centroid represents a cluster. After that, two procedures alternate [20]:

1. Assignment of samples to clusters: The euclidean distance between each sample and each centroid is measured. The sample is assigned to the cluster represented by the closest centroid.
2. Move of the centroid: The mean of all the samples attached to the same cluster is computed, the centroid is moved to this new position.

The two procedures alternate until no change in sample assignment is noticed any more. At that time, the sum of squared differences inside a cluster has reached a minimal value. Common issues of the k-means algorithm are the convergence to local minima and the fact that the hypothesis that clusters have spherical or ellipsoidal shapes is not always verified.

Hierarchical Clustering In hierarchical clustering, a tree of clusters, called a *dendogram*, is built [1]. All samples belong to the root cluster, and while descending down the tree, they are divided into different partitions according to some characteristics. Samples from the leaf clusters therefore share the characteristics of all their ancestor clusters. Data from different levels of granularity can be approached with this technique.

For example, the classification of the animal kingdom follows such a hierarchical classification [20], with the organization of different species into genus, then family, order, class, etc. Each cluster is included in upper level clusters, with the number of samples increasing while moving towards the top of the hierarchy.

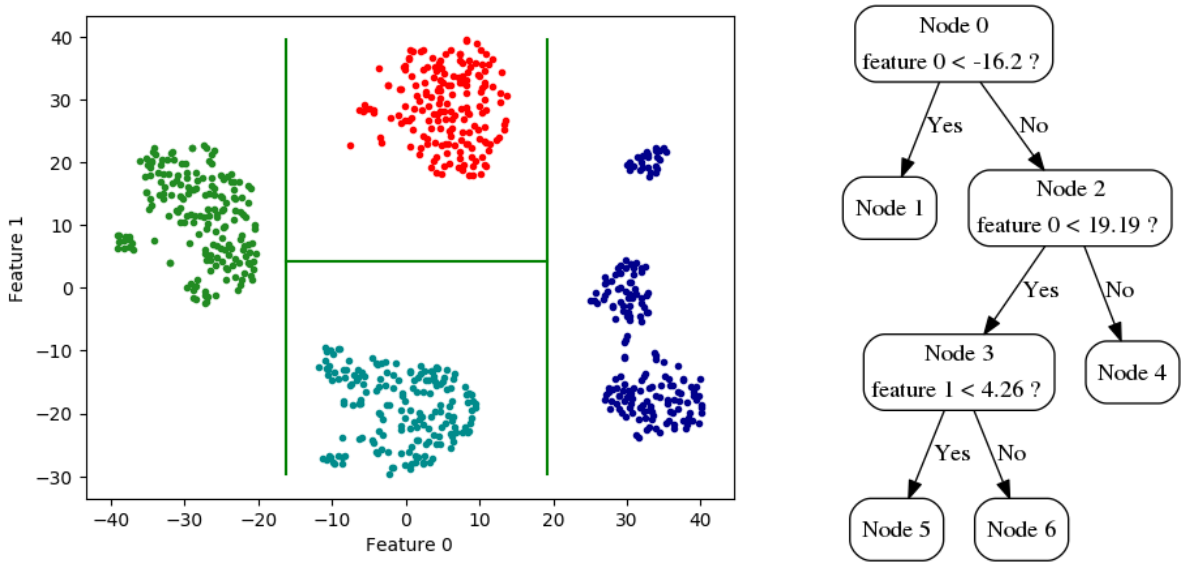
Hierarchical clustering can be divided into *agglomerative* (bottom-up) and *divisive* (top-down) methods [1]. Agglomerative methods start with single-sample clusters and merge the most appropriate ones to form new clusters, the same process is repeated until all samples belong to the same cluster. On the contrary, divisive methods successvely split a unique cluster into smaller ones, until a stopping criterion, e.g. a defined number of clusters, is reached. Divisive methods tend to be computationally less efficient, as explained in [20].

During the eighties, the euclidean distance between each pair of samples located in different clusters was commonly used as similarity measure. The exact computation of the measure could be either the average, the minimum or the maximum of those distances, depending on the papers [1].

Later, in the nineties, several well-known agglomerative algorithms were implemented:

- CURE (1998) [5]: A fix number of well-scattered points is chosen in each cluster as representatives of the cluster. Among all those representatives, the two closest are identified and the related clusters are merged. New representatives are then computed for each cluster and the operation is repeated until a pre-defined number of clusters is reached.
- ROCK (2000) [6]: The mechanism is the same except that instead of using the distance between clusters as similarity measure, the number of *links* is used here, with a link between a pair of points being the number of common neighbours. This algorithm allows to work with discrete values for the features.
- CHAMELEON (1999) [9]: A sparse graph is constructed with the k-nearest neighbour approach and a first set of small clusters is retrieved from it. Then, two clusters are merged if the inter-connectivity and the closeness of the two clusters gathered together is high compared to the internal inter-connectivity and closeness of the separated clusters.

Both k-means and the agglomerative hierarchical algorithms will be used later on in this thesis, as a source of inspiration or for validation of the developed algorithms.



(A) Data space with 4 clusters (B) Decision tree

FIGURE 2.1: With the decision tree (B), the data space is divided into regions. Samples from the same region are part of the same cluster and are then represented with the same color on (A): Node 1 (green), node 4 (navy blue), node 5 (light blue), node 6 (red).

2.2 What are Decision Trees?

At the beginning, decisions trees were developed to solve classification problems. Later, they were extended to perform clustering tasks.

2.2.1 Definition

Formally, a decision tree is used for classification and clustering, as it can fit a function with discrete output values. A decision tree consists of a root node, branches with regular nodes and leaf nodes. An example of such tree is shown on Figure 2.1. Each node of the tree proposes a test on a feature, while each branch under it specifies the possible values taken by this feature. Starting from the root, the features of the sample to classify are evaluated at each node and then guided to a certain branch following the test result, towards the leaf that will return the target of the sample [15].

Unlike other classification and clustering methods, decision trees only classify samples by looking on a subset of relevant features instead of the full set. This is a great computational advantage. Somehow, the test in each node of the tree can be seen as a conjunction of clauses, while each sample is a set of propositions [2] in first order logic.

Each sample is associated with a set of features. Those features can be seen as a vector, and therefore each sample is a point in the space having as many dimensions as the number of features; the *data space*. Decision trees create hyperplanes into the data space [20], dividing it into regions that ideally contain a single cluster. Figure 2.1 shows an example of such partition for a 2 dimensions dataset, a dataset where each sample has only 2 features.

Decision trees present many advantages:

- non-experts can easily understand them;
- they are robust to errors or outliers in the dataset [15];
- the training set can contain missing values for some features, in this case, the second best feature can be used to choose the best split [20];
- trees are compactly stored.

However, they also present disadvantages:

- the design of an optimal tree is difficult, especially if the cluster boundaries are complicated [20];
- the accuracy is limited because of the small number of considered features, only few among the whole set are used for a test in a node.

2.2.2 Construction of a Decision Tree

The basic algorithm (called ID3 for Iterative Dichotomiser 3) uses a top-down approach for the construction of the decision tree [15]. Each sample can be seen as a vector of different features and possibly a label giving the target of this sample. Each feature can either take a discrete or a continuous value.

At first, all samples belong to the root node. Each feature is considered separately and the samples are divided into partitions according to the different discrete values that the feature can take or according to the continuous value of this feature being above or below a threshold. A test then evaluates the quality of the partitioning, quantifying "how good is the split". Such a statistical test is performed for all features and for different thresholds in case of continuous feature values. The best test result determines what feature and what threshold are the most suitable for a split. The test can use the knowledge of the targets (classification) or not (clustering).

A branch is created for each of the possible values of the chosen feature (discrete feature) or based on the threshold (continuous feature). Then, the training set examples are split based on their value for the feature and assigned to newly created nodes. The same process is repeated with the subset of samples belonging to each new node, to define what should be the feature and the threshold selected for the next split [15].

When such a tree is constructed to fit the training set, it can overfit the data. Overfitting means that the tree classifies the training set so perfectly that it cannot be generalized to other data, and it fails to classify the test set. Sometimes, it is better to stop the growth of the tree before reaching a pure classification or to challenge the split on certain nodes after the construction.

2.3 Clustering Trees in the Literature

In the literature, only few papers take advantage of decision trees for clustering, those papers are presented in this section. Three relevant aspects of decision trees are highlighted to compare the different articles: The possible assignment of labels to data before building the tree, the applied split criterion during the construction of the tree and the improvement of the post-processing.

2.3.1 Label assignment

It has already been proved that decision trees are efficient for classification tasks, but for tackling the clustering problems, the targets are not available. One paper from the literature [13] therefore tries to reduce the clustering problem to a classification problem by assigning labels to the original data before building the tree. The authors assume that, besides the data points that form the actual clusters, the data space is uniformly distributed with non-existing points. Two targets are defined, the real points of the dataset and the non-existing points, so the clustering problem is changed into a classification problem and the newly assigned labels can be used in the computation of the split criterion.

2.3.2 Split criterion

To define a split criterion suitable for clustering, no pre-defined targets are available. A series of papers are based on the hypothesis that to obtain clusters, the intra-cluster distances (i.e. the distances between samples from the same cluster) should be minimized while the inter-cluster distances (i.e. the distances between samples from different clusters) should be maximized.

The most popular algorithm, CLUS, was developed by researchers from the Katholieke Universiteit Leuven in Belgium and from the Jozef Stefan Institute in Slovenia. Each split is done only if the variance is reduced when the node is split [2][21]. Several variations of the latter algorithm are available: The distance used in the variance computation can either be the regular euclidean distance [19] or a weighted version with weight decreasing with the depth of the node in the tree hierarchy [3]. The inter-clusters distance can be computed from the distance between prototypes, i.e. cluster representatives [2]. The intra-cluster distance can be derived from a measure of "mass concentration" [4] with normalization either on the total number of samples from the dataset or on the number of samples attached to a node.

When constraints are added to reflect the domain knowledge, the split criterion is not based solely on the variance of samples attached to a node. A global heuristic is computed to assess the global quality of the tree from the number of violated constraints [18].

In summary, the variance is the only split criterion present in the literature for clustering trees, derived slightly differently depending on the paper. This thesis will define other split criteria dedicated to clustering.

2.3.3 Post-processing

This section reviews different techniques found in the literature to improve the tree after its construction. It also addresses the question of leaves agglomeration.

Growth stop In classification, the growth of the tree is stopped when all the nodes are pure, i.e. all the samples in the node have the same target. However, the targets are not available in clustering, so other techniques must be used to stop the tree growth. In the clustering tree literature, in a paper in which the variance is used as heuristic for choosing the best split [2], the ratio of variance of the parent node to the sum of variances of the children nodes stops the growth if it reaches an arbitrary threshold.

Tree pruning After the construction of the tree, some splits are challenged and removed if they do not bring enough added value. This process is called pruning. Usually, the pruning is done to avoid overfitting or exaggeratedly large trees. The tree pruning can be done visually with the help of a dedicated user interface [13] or automatically. For the automatic method, a quality measure of the whole tree, such as regression error, can be computed. A split is removed if the quality measure is lower with the split than without it [18]. In this case, a validation set can help [2]. Similarly, a threshold can be defined and a split removed if the gain is under this threshold [16].

Pruning description is just given here in the interests of completeness. This technique is not applied in the scope of this thesis.

Leaves agglomeration After the final tree is built, each group of samples assigned to a leaf represents a cluster. However, it might happen that some clusters are divided into different leaves and for this reason a merge mechanism is applied at last. Indeed, if a label is assigned to each node, there will be many more labels than the actual number of targets.

Among the automatic methods, some papers such as [13] merge two clusters if they touch each other, in the sense that they meet in one dimension and intersect in all other dimensions. Some others measure the dissimilarity between sibling nodes from the distance between the samples of a leaf and the prototype of the other leaf [4]. Leaves are merged if the dissimilarity is under a pre-defined threshold. This method can also be applied to nodes that do not share the same direct ascendant. The method stops when the desired number of clusters is reached.

Agglomerative algorithms from Section 2.1.2 on hierarchical clustering tackle a similar problem.

CHAPTER

3

CONTRIBUTIONS

A good reason to work with decision trees is that they are lightweight in comparison with K-means algorithm for example. Practically, with K-means, predicting the cluster to which new samples belong requires to compute the distance between the sample and each centroid. This operation can be time-consuming for long features vectors. This thesis will focus on the use of decision trees to perform clustering, keeping in mind that so far in the literature, decision trees are mainly used for classification and only rarely used for clustering.

The goal of this thesis is twofold, in order to build a decision tree that defines to what cluster belongs a sample: On one hand, the discovery of new split criteria that do not request to know targets a priori and on the other hand the development of efficient leaves agglomeration mechanisms.

3.1 General Steps

The decision tree algorithm developed in this thesis follows the steps described at Figure 3.1. Each step will be conscientiously detailed in Chapter 4 on Methodology Evaluation.

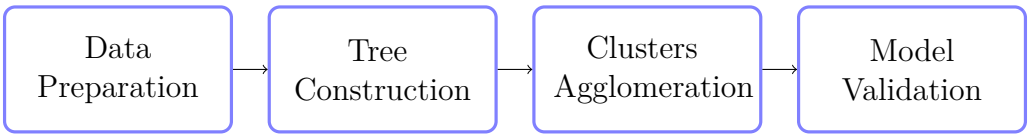


FIGURE 3.1: Algorithm steps.

Briefly, the data preparation step covers the import of the dataset and the possible normalisations applied to it. In this first step, the training and test sets are also created. The tree construction step builds the decision tree from the training set by applying the ID3 algorithm introduced at Section 2.2.2 and specific split criteria. At this stage, each leaf node of the tree represents a cluster and there are certainly more leaves than the number of actual targets in the dataset. Clusters referring to the same targets must be merged together, this is done in the specifically implemented clusters agglomeration step. Finally, in the validation step, metrics are computed to assess the performance of the tree to classify the test set.

The contributions of this thesis focus on two axes: The tree construction with the introduction of new split criteria and the clusters agglomeration with improved clusters merging techniques.

3.2 Tree Construction

The ID3 algorithm is used for the construction of the tree and it requires a specific split criterion to be chosen. Split criteria can be divided in two categories: Those who require the knowledge of the targets a priori and those who do not. For clustering, only the second category is relevant and so the newly implemented split criteria are part of it. However a split criterion from the first category will still be presented as it is later used as a reference, for comparison purposes.

The word *impurity* in the following paragraphs refers to the value of a split criterion for a specific partition of the data. A low impurity indicates that the samples are really similar to each other and is therefore desirable when trying to find the best split for a node.

3.2.1 Cross-Entropy

This split criterion can only be used in supervised learning as it requires to know the targets, the classes, in advance. It is part of the first category of split criteria mentioned here above.

In classification, a typical test to define which feature to select is the information gain [15]. It reflects the effectiveness of a feature to partition the training data. In scikit-learn library [17] and in this thesis implementation, this test is based on the cross-entropy as a measure of the impurity of the data. The cross-entropy reflects the possibility to wrongly classifying a sample. In this sense, the best feature according to the information gain is the feature that decreases the cross-entropy when it is used for partitioning

$$\text{CrossEntropy}(S) = -\sum_{k=0}^{K-1} c_k \log_K(c_k)$$

where c_k is the proportion of class k in node S (i.e. the number of samples of class k divided by the total number of samples in node S) and K is the number of possible classes.

The gain along a feature f [15] is defined as

$$\text{Gain}(S, f) = \text{CrossEntropy}(S) - \sum_{v \in \text{Values}(f)} \frac{|S_v|}{|S|} \text{CrossEntropy}(S_v)$$

where Values(f) are all the possible values of the feature f.

3.2.2 Box Volume

This criterion has been developed in the scope of this thesis. The objective is to localize clusters by contouring them with a box. Empty regions will not be considered anymore if they are not part of a box, so one step at a time, the considered volume in the data space will get smaller. Figure 3.2 shows the shape of such boxes in two dimensions.

According to ID3 algorithm, when splitting a node from the tree, different thresholds are compared, for each feature. Each threshold can somehow be seen as a cut of the data space, creating two partitions. The impurity of the parent node is computed, as well as the impurities of each partition.

Once the partitioning is made with a threshold on a specific feature, the center of each partition is computed as the mean of all samples in the partition. For the mean computation, all features are considered, not only the feature used for the cut. Separately for each feature (i.e. each dimension of the data space), starting from the center, the box width is increased incrementally until at least 95% of the samples are located in the box, as shown on Figure 3.3. The impurity is the product of all box widths in each dimension. 95% is an arbitrary parameter of the algorithm and this value could be discussed. The only important point is to ensure the robustness to outliers with a number that is lower than 100%.

By construction, this impurity does not reflect the volume of the box containing exactly 95% of the samples, as the width of each dimension is computed separately. The computational cost of refining the box shape to include exactly 95% was too high and did not present any advantage for the tree construction. For this reason, it has not been kept. The final *box volume* criterion computation is

$$BoxVolume = \text{volume of the box containing 95\% of samples along each dimension.}$$

When the sum of the boxes volumes from both partitions (weighted by the number of samples in each partition) is smaller than the volume of the initial partition box, empty volumes of data space are removed, and so the tree construction is going toward the right direction.

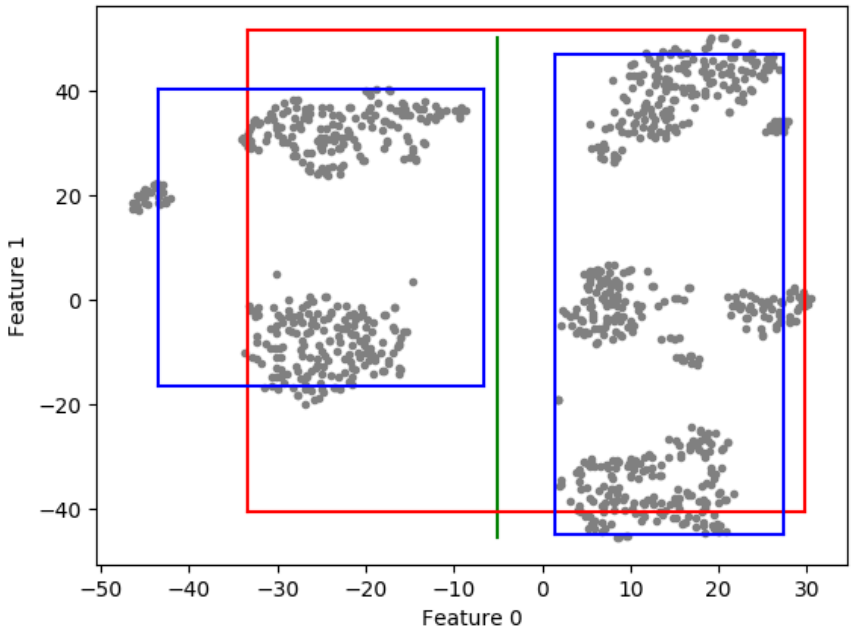


FIGURE 3.2: The *box volume* criterion is applied to assess whether the threshold on feature 0 (green line) is a good choice for splitting the node. The impurity before the split is computed as the area of the red box. After the possible split, the impurity of each partition is computed as the area of the blue boxes.

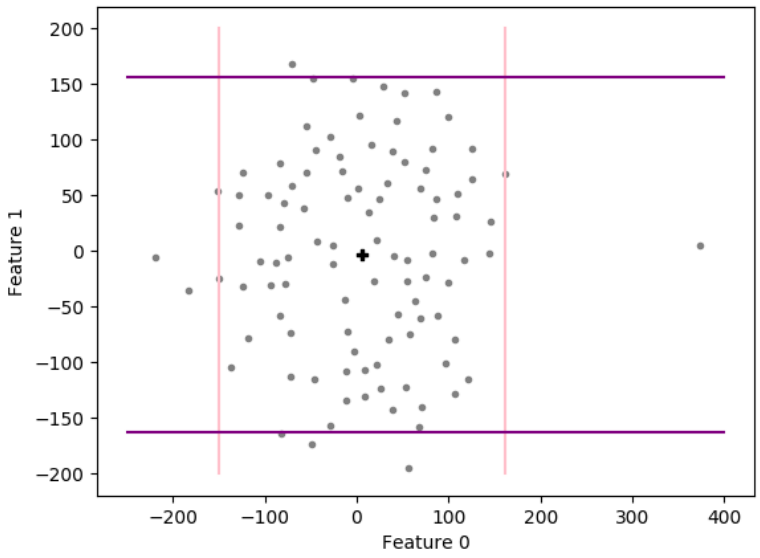


FIGURE 3.3: Box is grown from the center (black cross). For feature 0, at least 95% of the samples are located between the two pink lines (x-axis) and for feature 1, at least 95% of the samples are located between the two purple lines (y-axis). The box volume, in this case the area, is the product of box widths in each dimension.

3.2.3 Graph Closeness

This criterion has been developed in the scope of this thesis as well. It is inspired by the relative closeness definition from CHAMELEON [9] but this latter is used to merge different clusters, not to split, as CHAMELEON is a hierarchical agglomerative algorithm (see Section 2.1.2). However, the computation is CHAMELEON returns a unique value to characterize a split, based on the impurity of the parent node and both the children nodes. In this thesis, the split criterion reflects only the impurity of a single child node.

A graph is created from all the samples of the node to split. Each sample is linked to its nearest neighbours according to the euclidean distance. The weight of each edge is the distance itself, meaning that two samples close to each other are linked by an edge of low weight, while on the contrary two samples far from each other are either linked by an edge of high weight or not linked at all. Such graph is shown on Figure 3.4A.

The cut creates two partitions and the impurity is computed separately for each. All the edges linking samples from one partition to the other are removed from the initial graph, as well as the edges between samples that do not belong to the current partition. The weights of the remaining edges are summed. The impurity is computed as the inverse of this sum of weights

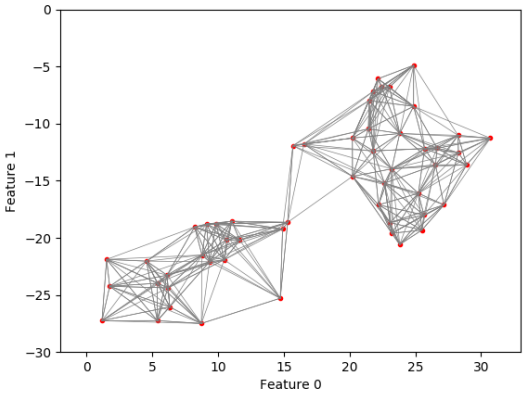
$$GraphCloseness = \frac{1}{\sum_{e \in E} weight(e)}$$

where E is the set of edges in the graph after the removal.

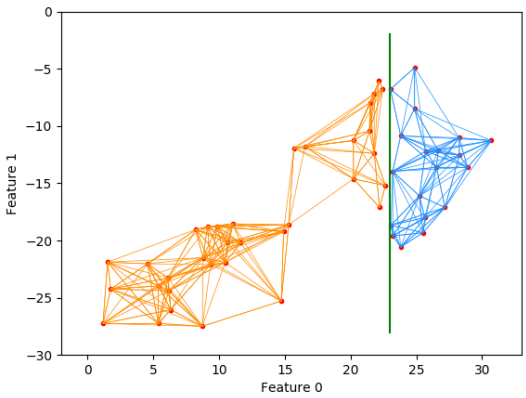
With the *graph closeness* criterion, when a cut is done in the middle of a cluster (Figure 3.4B), many edges will be removed from the graph. As a consequence, the sum will be lower and the impurity higher, indicating a wrong choice of threshold for a cut. On the contrary (Figure 3.4C), a cut located between clusters will not remove many edges and the impurity will be lower.

A notable difference between this criterion and the two previous ones is that it requests the knowledge on the samples assigned to the node before the split, through the existence of the graph, and cannot be solely computed based on the partition samples.

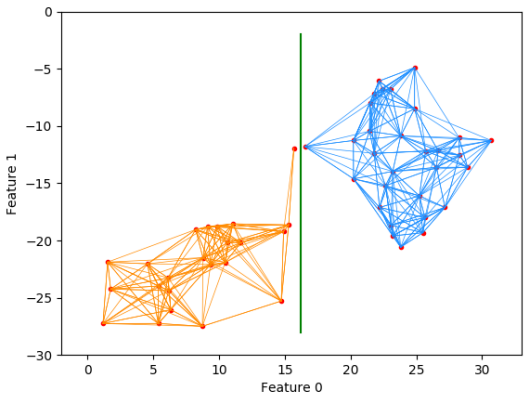
Another related criterion was investigated then abandoned in the scope of this thesis: Instead of computing the inverse of the weights sum, the impurity was the mean of the edges weights. Indeed, with such criterion, a cut between two clusters will be encouraged as it will remove high weight edges and so decrease the mean. On the contrary, a cut in the middle of a cluster will remove edges with low weight and increase the mean. Unfortunately, this criterion did not successfully find good splits because only few edges, almost none actually, link two separate clusters for small a number of nearest neighbours. The need of a high number of nearest neighbours has arisen, requesting high computational cost for equivalent results.



(A) Original graph



(B) Bad split



(C) Good split

FIGURE 3.4: For each partition, the impurity is computed as the inverse of the sum of edges weights, in a graph where edges not connected to samples of this partition are removed. On those two figures, edges participating in the computation of the impurity criterion are represented in orange for the partition to the left and in blue for the partition to the right.

3.3 Clusters Agglomeration

After the construction of the tree, each leaf node represents a region in data space where a possible cluster is located. For clustering, merging those leaf nodes is critical as the tree does not stop its growth when reaching nodes only containing samples from a unique target, and so most clusters are divided. This section will describe how clusters are merged and how labels are assigned to them.

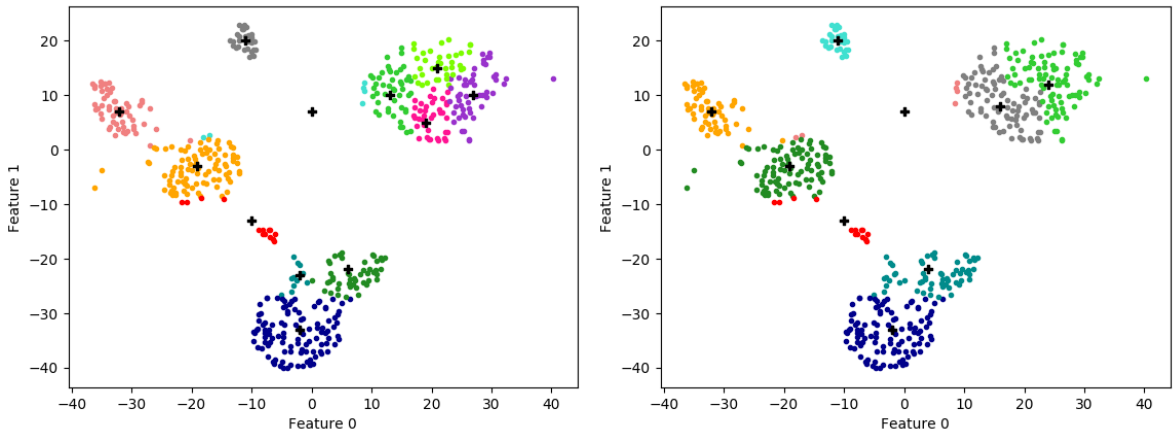
3.3.1 Based on the Labels

Classification decision tree algorithm from scikit-learn library [17] is part of the supervised learning algorithms. In this case, samples from each node are assigned the label of the most represented target in the node. While this technique can be used as a reference, it cannot be applied to clustering as the target labels are not available.

3.3.2 Prototypes

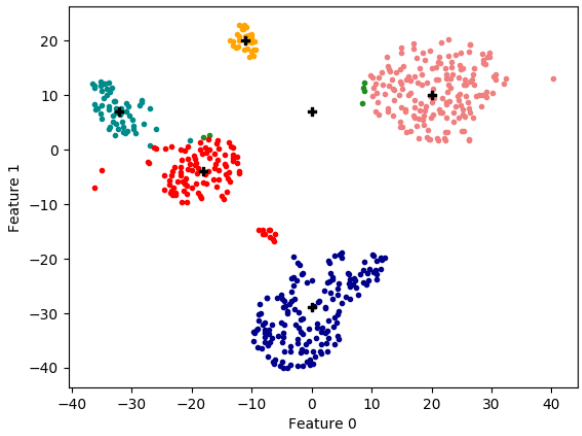
Single Prototype For each node, the mean of the samples is computed and is used as a prototype, i.e. representative, of the node. The two nodes having the closest prototypes in terms of euclidean distance are merged. Then, the prototypes are re-computed with the new samples groups. These two steps are repeated until a pre-defined number of prototypes is left: The number of targets in the dataset. Each remaining cluster is assigned a label. Figure 3.5 presents four iterations of this algorithm.

Set of Prototypes Similarly to the previous technique, another implemented technique consists of representing each node with several prototypes. In CURE algorithm [5] from section 2.1.2, a set of samples is randomly chosen among the node samples. Then, to limit the effect of outliers, those are moved towards the center, i.e the mean of the cluster, by a fraction α. After this move, among all the prototypes from all the nodes, the two closest based on the euclidean distance are identified and the related clusters are merged. The prototypes are re-calculated for all nodes and the same operation is iterated until a pre-defined number of clusters is left, the number of targets. This mechanism introduces a better flexibility in the shape of the clusters. The only difference between the implemented algorithm and CURE is that CURE chooses the new prototypes as being the farthest samples from the previously chosen prototypes, and this is not the case of this thesis algorithm.

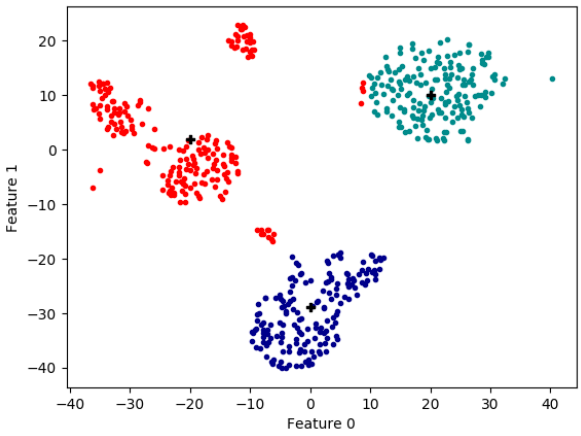


(A) 12 prototypes

(B) 9 prototypes



(C) 6 prototypes



(D) 3 prototypes

FIGURE 3.5: Four different iterations of the *single prototype* agglomeration algorithm applied to 3 clusters. Each node is represented with a prototype (black cross) and the samples attached to a node are displayed in the same color. The nodes with the two closest prototypes are merged and new prototypes are computed. This process iterates until the number of prototypes equals the number of targets.

3.3.3 Box Distance

As previously stated, after the tree construction, there are more nodes than the actual number of targets and each node has a set of samples attached to it. Similarly to the *box volume* split criterion from Section 3.2.2, a box can be constructed for each node, which contains 95% of the samples along each dimension.

The basic idea of this agglomeration mechanism is to merge the two nodes whose boxes are closest to each other, in terms of euclidean distance between borders. After the agglomeration, the boxes are re-computed based on the new set of samples and the process is repeated until the number of targets is reached. Such process is illustrated on Figure 3.6.

To measure the distance between two boxes, an accumulator *Acc* is used. It is a symmetric matrix of size $N \times N$ where N is the number of nodes. $Acc[m, n]$ is the distance between node m and node n boxes. Leaf nodes are picked in pairs and the *Acc* value is increased by a certain amount while iterating over each feature

$$Acc[m,n] = \sum_{f \in F} \begin{cases} (box_{inf,m,f} - box_{sup,n,f})^2 & \text{if} \quad box_{inf,m,f} > box_{sup,n,f} \\ (box_{inf,n,f} - box_{sup,m,f})^2 & \text{if} \quad box_{inf,n,f} > box_{sup,m,f} \\ 0 & \text{otherwise, i.e. boxes overlap} \end{cases}$$

where

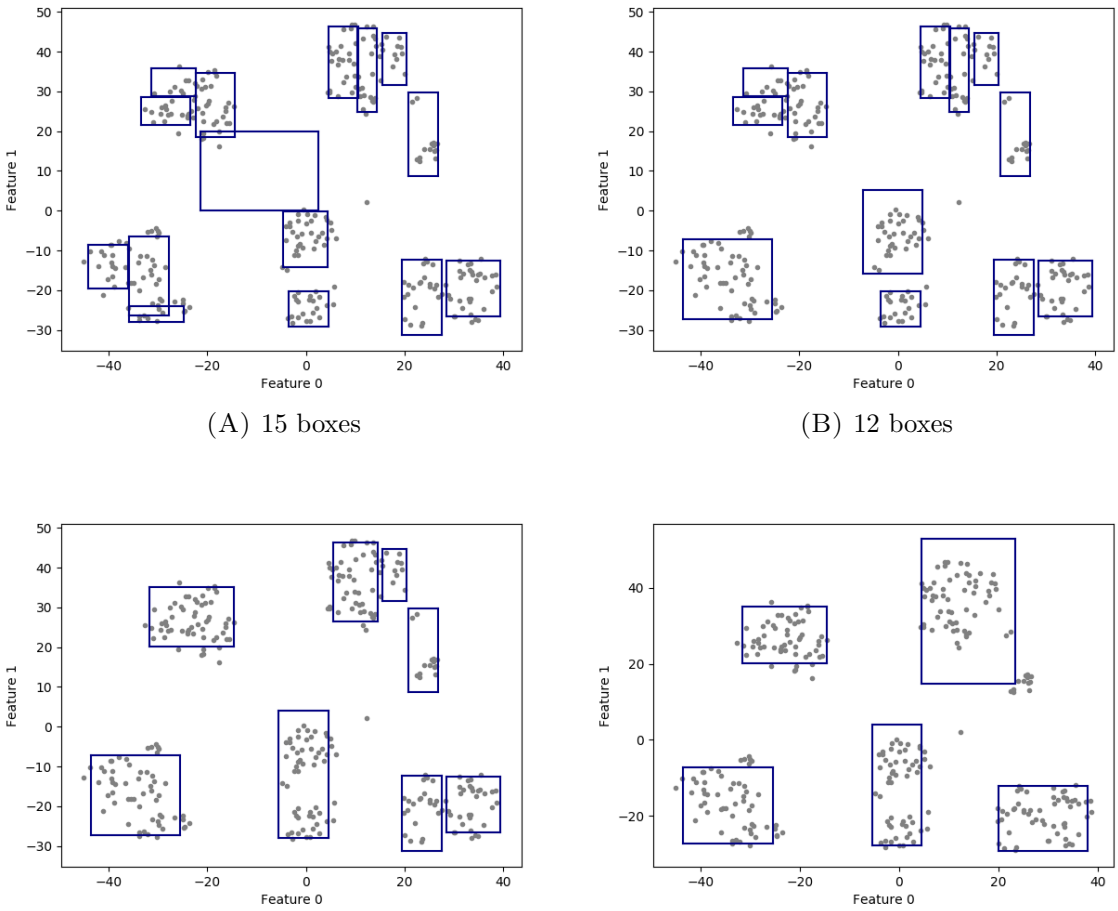
- F is the set of all features, the number of dimensions of the box;
- $box_{inf,m,f}$ and $box_{inf,n,f}$ are respectively the inferior limit of the box of node m and n along feature f;
- $box_{sup,m,f}$ and $box_{sup,n,f}$ are respectively the superior limit of the box of node m and n along feature f.

Once the accumulator is constructed, the minimum of the matrix is found and the index of this minimum indicates which nodes will be merged.

3.3.4 Graph Connectivity

The *graph connectivity* agglomeration mechanism is based on the intuition that actual clusters are well-connected. In this thesis, this connectivity is represented by the edges on a graph of nearest neighbours. Each possible pair of leaf nodes is picked, and the number of links going from samples from the first node to samples from the second node, or in the opposite direction, are counted. The two nodes with the highest number of links are merged and the process it iterated until the number of wanted clusters is reached. This is illustrated on Figure 3.7.

The number of links between two clusters is also stored in a symmetric matrix of size $N \times N$ where N is the number of nodes. The two nodes at the index of the maximum value are chosen to be merged.



(A) 15 boxes

(B) 12 boxes

(C) 8 boxes

(D) 5 boxes

FIGURE 3.6: Four differents iterations of the *box distance* agglomeration algorithm applied to 5 clusters. Each node is represented by a blue box. At each step, the two nodes whose boxes are closest to each other are merged until a fix number of clusters is reached.

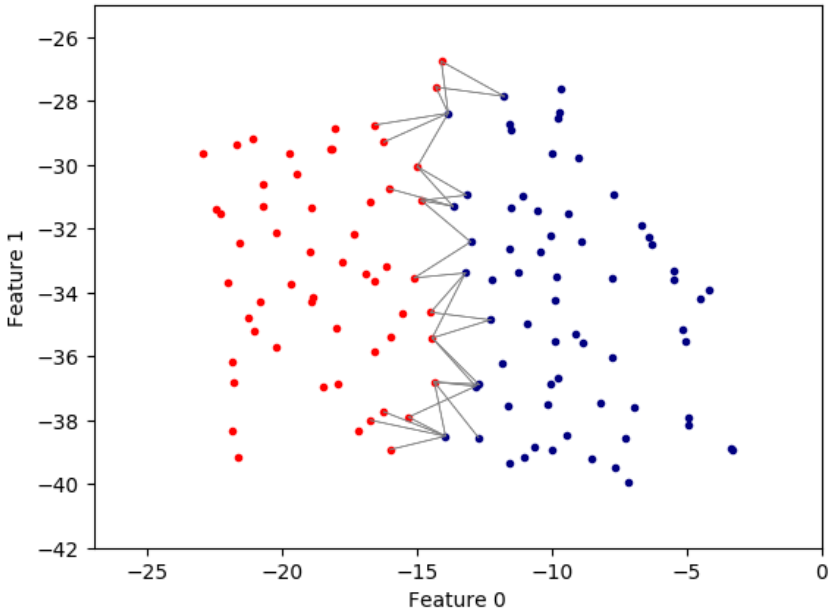


FIGURE 3.7: In the *graph connectivity* agglomeration algorithm, a nearest neighbours graph is constructed. Then, the number of edges in between samples from two different nodes (blue and red) is counted. A high number of links encourages the merge of those two clusters.

CHAPTER

4

METHODOLOGY EVALUATION

In this chapter, the different steps of the alorithm developed for this thesis are detailed one by one, following Figure 4.1 that was already shown previously.

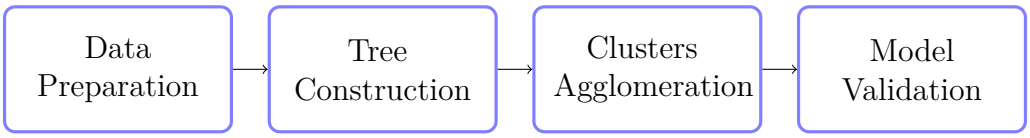


FIGURE 4.1: Algorithm steps.

To take advantage of the powerful scikit-learn [17] tools, this thesis is implemented using Python language. `Numpy` arrays are used as the main way of handling the data and all graphs are generated with `matplotlib` library. But not everything is available without any effort, so several tools are also implemented to answer this thesis needs, among them is the necessity to visualize the decision trees: `Graphviz` library, implementing DOT language, has been put to good use to create nodes (with HTML code) and edges, to form handsome figures.

In this chapter, several decision trees are shown. It is important for the reader to understand the meaning of such figures. So, the first section of this chapter will give an overview, a sort of legend, of what a decision tree looks like. Then, going to the heart of the matter, the algorithm steps will be browsed one by one as promised.

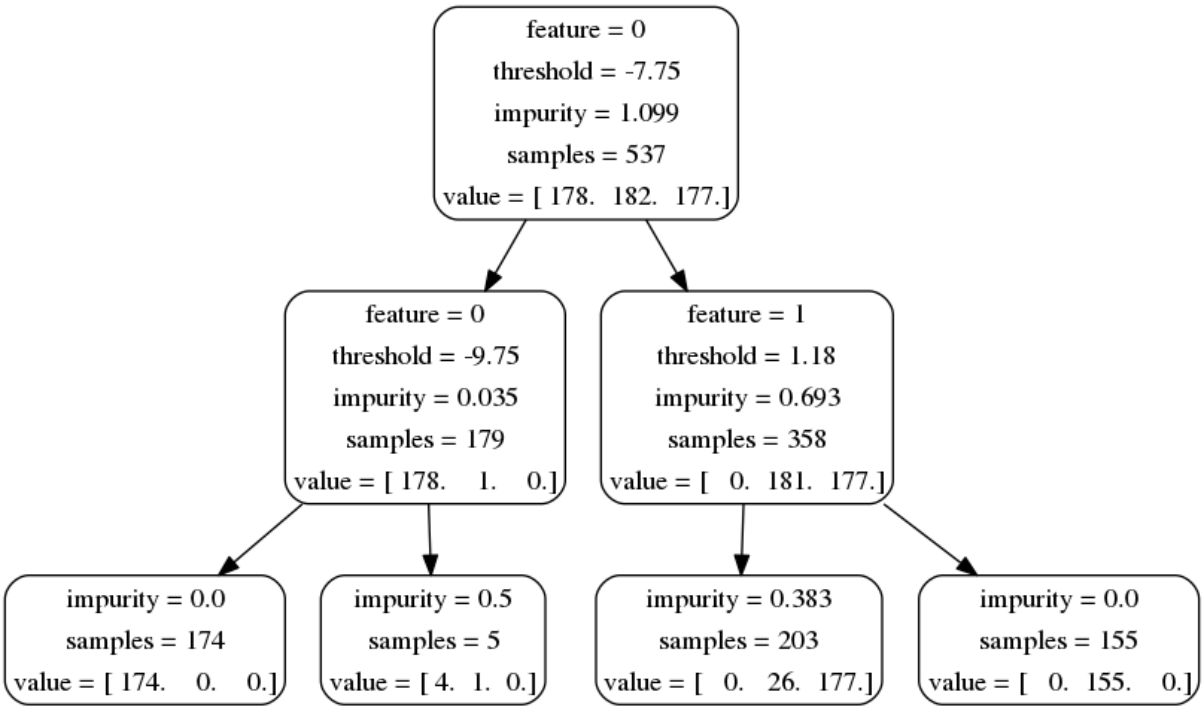


FIGURE 4.2: Example of decision tree.

4.1 Decision Tree Legend

Figure 4.2 shows an example of the generated decision tree. This figure displays a tree of maximal depth 2. The root node, the summit of the tree, is located at depth 0. The children are located at depth 1 and the grandchildren, i.e. the leaves, are located at depth 2. This maximal depth is used to stop the growth during the tree construction: Nodes are not split any more once they have reached the maximal depth.

Each node of the tree contains a number of *samples*. On the tree from Figure 4.2, the root node contains 537 samples. After a first split, samples are divided according to the test result and the sum of depth 1 nodes samples numbers (179 + 358) equals the root node sample numbers (537).

The keys *feature* and *threshold* give the detail of the test applied in this node. In the root node of our tree example, the test was "is the feature 0 value under -7.75?". If the answer is Yes, the sample is guided to the left child node. Otherwise, it is moved to the right child node.

The node *impurity* returns the value of the chosen split criterion for this node. The lower the impurity, the most similar the node samples are. The *value* field gives the number of samples of each class in the node. It is not used at any time in the algorithm but it gives a view on the performance of the clustering. The vector has a length of 3, which means that 3 clusters are expected to be discovered with the tree.

4.2 Data Preparation

The first step of the algorithm implemented in this thesis is the *data preparation*. It consists in the selection of a relevant dataset, a possible dimensionality reduction or make it suitable for the experiments and the creation of the training and test sets.

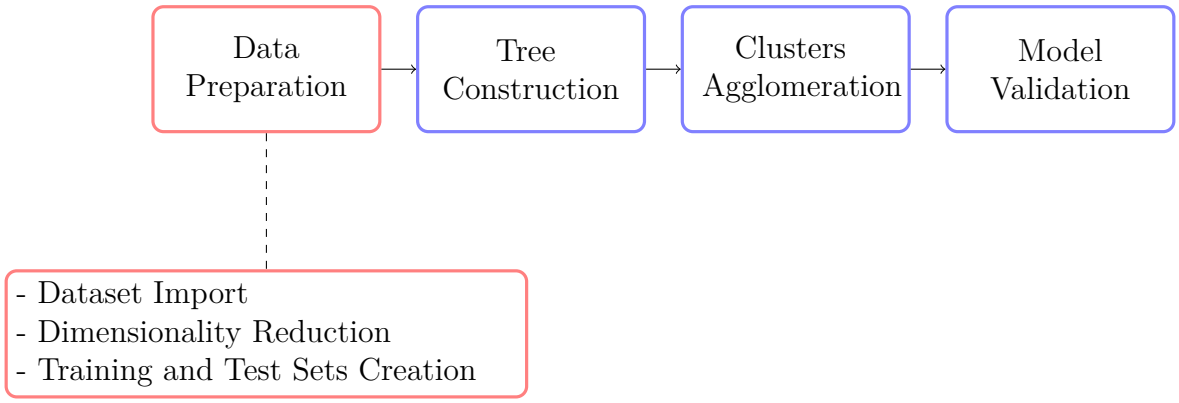


FIGURE 4.3: Algorithm steps: Data Preparation.

4.2.1 Dataset Import

Several datasets exist in the scikit-learn library [17]. Three of them were used in the scope of this thesis and are presented in this section: Newsgroups, digits and MNIST. It is also possible to select only a subset of targets to work with easier datasets.

NEWSGROUPS

This dataset [10] gathers newspapers articles and emails on 20 different topics such as atheism, baseball or space. An example of such article is shown on Figure 4.4. Those topics will be the different targets. In this document, the three sub-mentioned targets were used for the analysis, and they were divided into the training and the test sets with two thirds of the data dedicated to the training set and the rest to the test set (1670 samples in training, 1110 samples in test set).

Pre-processing To build the tree, each sample, i.e. each newspaper article, must be represented by a vector of different features. Those features can be compared in each node to define which one is the best to split the data with a threshold. For document analysis, the features describe the occurrence of a specific word in the articles.

However, to obtain an occurrence vector with more relevant information than the plain occurrences, a scikit-learn tool called `TfidVectorizer` is used. Tfid stands for *Term Frequency Inverse Document Frequency*, a technique used in information retrieval from documents. The occurrences are weighted and higher importance is given to words that do not appear often in the whole set of articles so they might reveal good information on this article specific content. By setting the parameters `stop_words='english'`, words

```
In article <1993Apr2.184338.18205@dvorak.amd.com> twhite@mozart.amd.com (Tom
White) writes:

>    The highest single-game attendance was Game 5 of the 1959 World Series,
> October 6, at the LA Coliseum.  White Sox over Dodgers, 1-0.
>
>    Gate?  Officially 92,706.
>
>    Largest regular-season game?  78,672, again in LA, for the first
> game in the City of Angels -- Opening Day, April 18, 1958 (home opener,
> anyway).
>
>    The Rockies might really nail the record.
>
>    The record attendance for a doubleheader is larger, but since dh's are
> all but nonexistent nowadays, why bother listing it...

Wasn't there an 85,000 New York at Cleveland game in the late 40's?

jhon rickert
rickert@nextwork.rose-hulman.edu
prediction for 1993:  Marlins: 70 wins, Rockies: 50 wins
```

FIGURE 4.4: Example of newsgroup dataset - article 102595 from category `rec.sport.baseball`.

that are common in English language, such as "the", "sometimes", "between" are ignored. Similarly, with `strip_accents='ascii'` parameter, accents are not taken into account. A total of 34 449 different features were identified.

Strengths & weaknesses The main problem encountered with this dataset is the difficulty to get a sense from the tree. To illustrate with Figure 4.5, on each node is shown the word used for the split as the feature, but it is challenging to understand why this word was chosen: What does a split on the words "agree" or "official" mean? What direction is the tree taking? What targets are we discriminating here? For this reason and because of the huge number of features, this dataset was not used during the development phase.

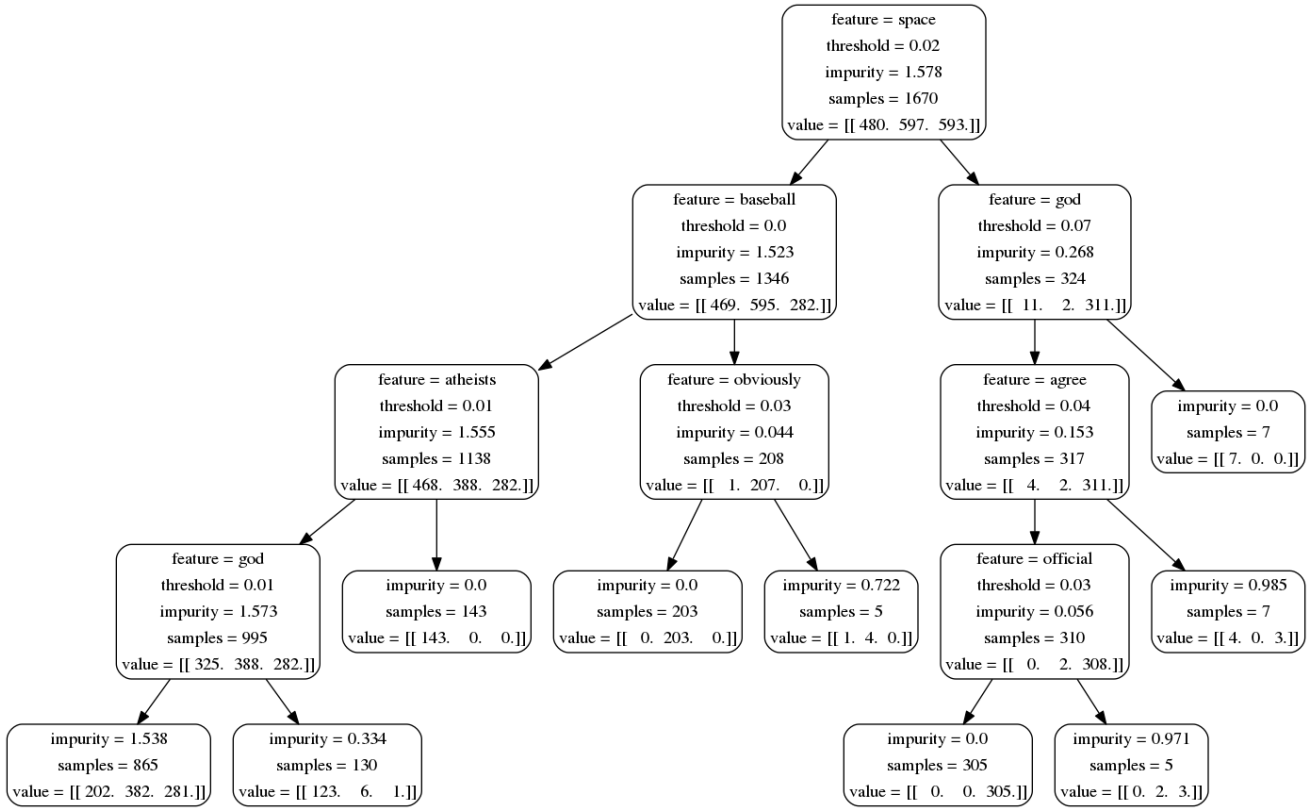


FIGURE 4.5: Newsgroups dataset gathers articles on different topics. The feature vector of each sample consists of the weighted occurrence of different words. The decision tree on the figure was built from those feature vectors for the topics “space”, “atheism” and “baseball”.

DIGITS

Around 1800 handwritten digits images are separated in 10 classes, the targets. All samples figure numbers ranging from 0 to 9. The images size is 8 by 8 pixels. The dataset comes from UCI Machine Learning Repository [12].

FIGURE 4.6: Samples of the digits dataset (8x8 pixels)

Pre-processing The image, as a matrix, will be reshaped to form a vector, in which each feature represents the intensity, i.e. the grey level, of a single pixel. This vector will be the input of the decision tree. On each node from Figure 4.8, the pixel used for the split is represented in red. On the child nodes, this same pixel is represented either in white if the value was less than the threshold or black otherwise.

Strengths & weaknesses A split following a pixel grey level is easy to interpret in the tree if we can visualize what pixel is used for the current split but also what pixels were used previously. By descending into the tree, the shape of the different numbers is slowly getting more obvious, as seen on Figure 4.8. Sadly, picking 5 or 6 pixels into the image is not sufficient to define the target of the image, 10 to 20% of the test set images are wrongly classified.

MNIST

MNIST stands for Modified/Mixed National Institute of Standards and Technology, an agency of the US Department of Commerce. Similarly to the digits dataset, the MNIST dataset [11] gathers 70 000 handwritten digits divided into 10 classes, the targets. The images size is 28 by 28 pixels.

FIGURE 4.7: Samples of the MNIST dataset (28x28 pixels)

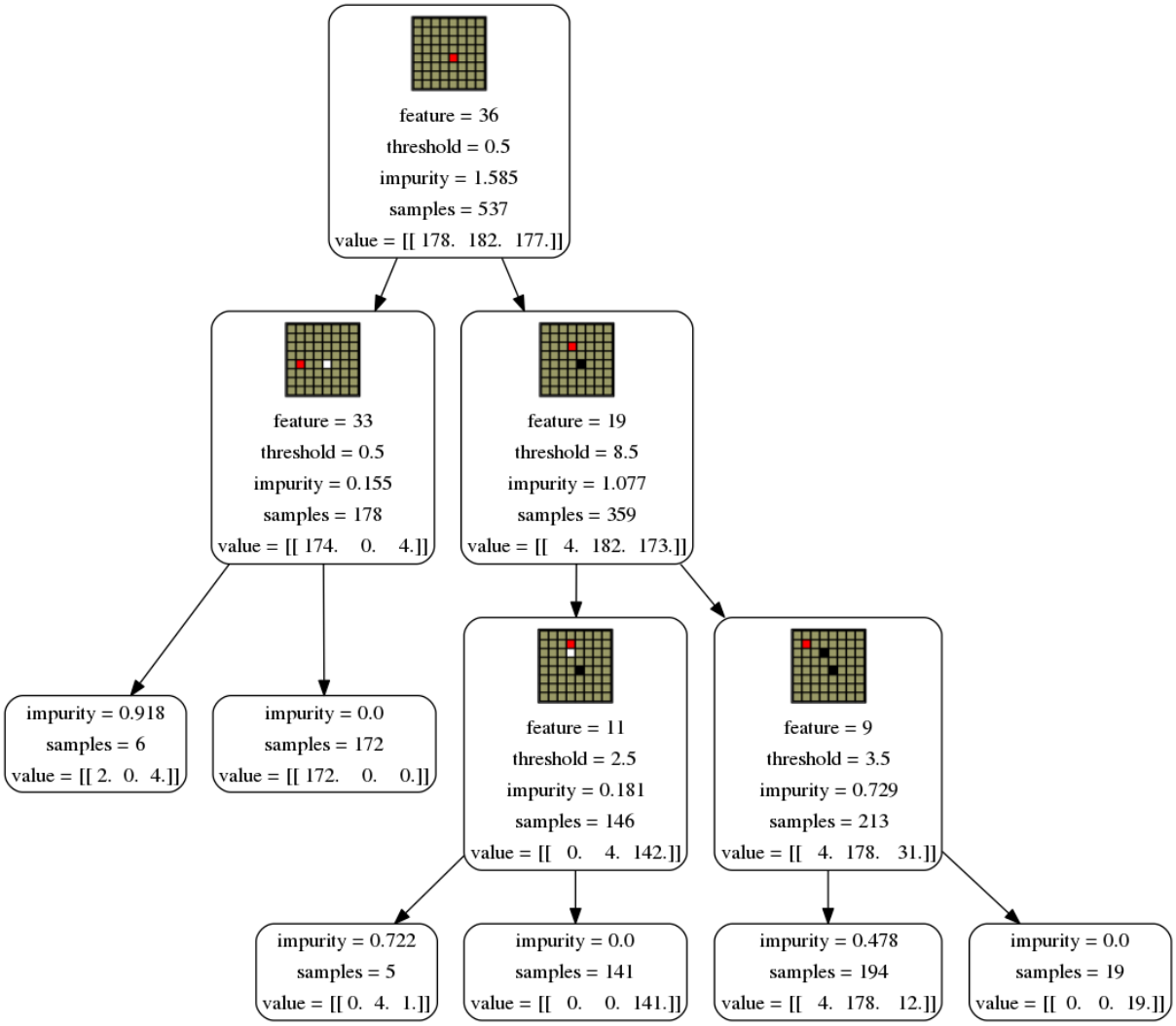


FIGURE 4.8: Digits dataset: The features used to build the decision tree are the pixel intensities. The number next to the feature label is the pixel id and the grey square represents the pixels of the image. The red pixel is the pixel currently used for the split test. Black and white pixels respectively indicate if a previous split led to the right or left branch.

Pre-processing The pre-processing is similar to the digits dataset, except that the number of features is increased to 784 due to the larger number of pixels.

Strengths & weaknesses As each vector has 784 features, a tree with a depth of 5-6 is not sufficient anymore to discriminate the target of a sample. To get better results, a tree of a depth of at least 20 is necessary. Such a tree is difficult to read because it is really large and consists of many nodes. Unfortunately, just by taking a vector of grey levels without any further pre-processing of the images, the misclassification rate is similar to the digits dataset and is not good enough to be used as a reference as such.

4.2.2 Dimensionality Reduction

The high misclassification rate found with digits or MNIST datasets is caused by the dimension of the vector representing each sample. Indeed, counting only on the intensity of 5-6 pixels is not sufficient to recognize the represented digit, especially when the number of pixels is great. To overcome this difficulty, the dimensionality of the data is reduced and each sample is now a data point in a 2 or 3 dimensions space. Luckily, the chosen datasets form nice clusters in such low dimension spaces. For a classification task, this processing decreases the misclassification rate down to 3% for digits and around 7% for MNIST datasets with t-SNE technique, detailed here below.

However, a disadvantage of the dimensionality reduction is the loss of interpretability of the data. This is not an issue for this thesis as the main concern is to discriminate well-separated clusters without human interaction, no matter what the data represents.

Two well-known algorithms were compared to reduce the dimensionality of the data: t-SNE and PCA.

t-Stochastic neighbour Embedding SNE, standing for Stochastic neighbour Embedding, is a dimensionality reduction technique that tends to preserve the neighbourhood of data points when going from the high to the low dimensional space [8]. A Gaussian is centered on each data point and it helps to define the probability distribution over the neighbours of the point. When two points are close in the high dimensional space, the probability will be high, and when they are widely separated, the probability will be low. The cost function of the algorithm ensures that high or low probabilities distributions in high dimensional space are preserved in the low dimensional space. However, SNE algorithm tends to pack the data points in crowd and the cost function is asymmetric: Nearby points represented far away or separated points represented really close do not lead to the same cost.

t-SNE is derived from SNE and overcomes its main drawbacks [14]. Instead of a Gaussian, a Student distribution is used. It also results in faster computation as the exponential of the Gaussian is removed so the cost function is simpler and symmetric.

Using t-SNE for this thesis allows to reach better performance for the classification tasks on both digits and MNIST datasets and to easily represent the clusters in 2 or 3 dimensions on a plot. The t-SNE algorithm is available from scikit-learn library [17], in the `manifold` package.

However, dimensionality reduction is time-consuming so for daily work with more than 2000 samples, a mechanism has been put in place to save the data into a text file and retrieve it when needed. It also allows to compare different split criteria on the exact same data.

Principal Component Analysis PCA is another algorithm that can be used to reduce the dimensionality of the data. It tends to find correlation in the data and apply orthogonal transformations to represent it using principal components. The first principal components present a lot of variation of the data, while the last ones barely vary.

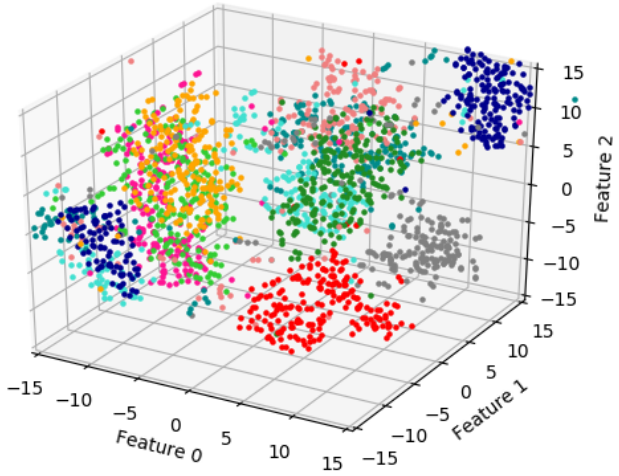


FIGURE 4.9: Dimensionality reduction technique t-SNE is applied on MNIST dataset (3 components). On this figure, the colors refer to different targets.

By focusing only on the first principal components, dimensionality of the data can be reduced without major losses of information. PCA can also be applied on the digits or MNIST datasets.

4.2.3 Training and Test Sets

The dataset is divided in two groups: The training set and the test set. The samples from the training set are used to construct the tree with the ID3 algorithm and to agglomerate the leaves afterwards. Then, to assess the performance of the tree in finding clusters, the test set samples are guided through the tree and the predicted target can be compared with the actual one. More details on the validation will be covered in Section 4.5. With the test set, overfitting can be discovered.

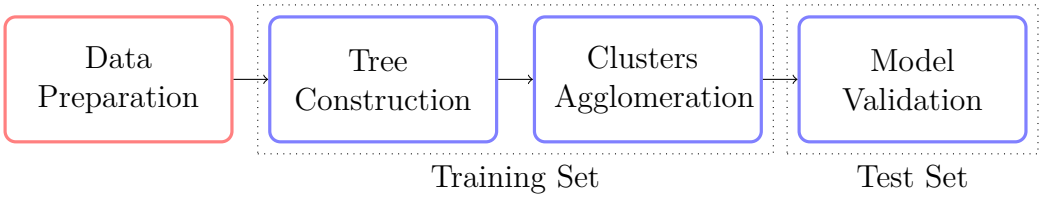


FIGURE 4.10: The training and test sets are used at different moments in the algorithm. The training set is used during tree construction and clusters agglomeration, while the test set is used for the model validation.

In this thesis, two thirds of the samples are dedicated to the training set, while the last third remains for the test set. However, to guarantee that the results are not biased by this arbitrary partitioning, experiments are repeated 30 times, with a random partitioning each time, but still keeping this ratio two thirds - one third. In the literature, this process is called *cross-validation*.

4.3 Decision Tree Construction

Once the data is prepared, the algorithm goes to the heart of the matter: The construction of the decision tree. The maximum depth of the tree is a parameter, and the split criterion has to be chosen.

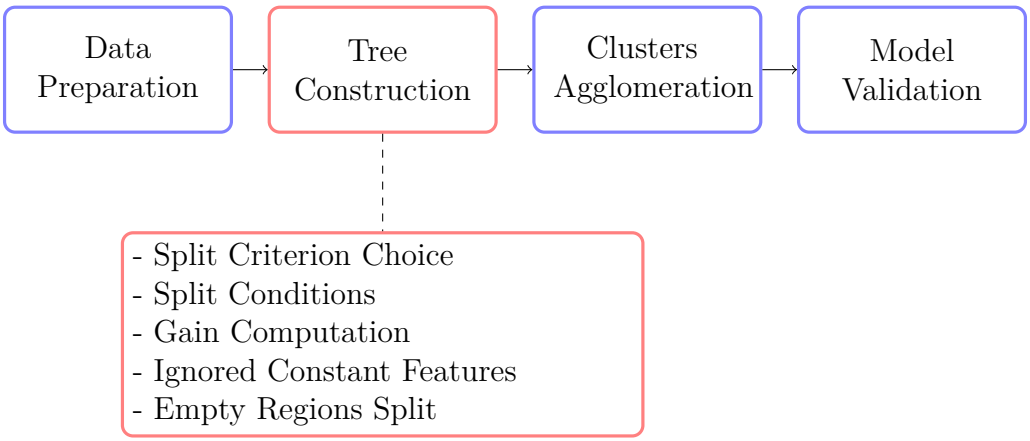


FIGURE 4.11: Algorithm steps: Tree Construction

4.3.1 Scikit-learn

In the early stage of this thesis, trees were constructed with `DecisionTreeClassifier` object from the scikit-learn library [17]. All parameters were set to their default value except for the choice of the split criterion and the maximum depth of the tree. Cross-entropy criterion is already part of the library. The maximum depth of the tree has a huge impact on the accuracy of the future predictions as a wrong setting of this parameter is taken for responsible in case of under- or over-fitting of the training set data.

However, the criteria implementation in scikit-learn library was designed for optimization and was therefore technically not easily modifiable. It was written in Cython, a language that takes the advantage of C compilation mechanisms to apply them to Python, introducing many constraints on the available libraries or the possible datatypes. Additionally, it does not bring the required flexibility for this thesis, in the sense that it was only possible to develop a new criterion based on the node samples and not on the whole batch of initial samples for example.

4.3.2 Improved ID3 Implementation

For the reasons explained above, the scikit-learn tree will not be used and a basic ID3 algorithm without post-pruning was fully re-implemented besides the framework. All the results presented in this document are generated with this version of the tree classifier. For cross-entropy criterion, trees generated with scikit-learn tool or with this re-implementation give the exact same features and thresholds for splits. Scikit-learn implements the tree construction on a depth-first manner, while this algorithm uses breadth-first, but this is just a representation, it has no impact on the final result.

Split Criterion Choice The criteria from Section 3.2 were all implemented, taking advantage of the flexibility of the new implementation. The k-nearest neighbours graph from the *graph closeness* criterion was created with the scikit-learn library. For *Graph Closeness* criterion, the number of nearest neighbours is set to 10 times the number of features.

Split Conditions The only conditions for a valid split are that the maximum depth of the tree has not been reached yet and that each child node contains at least 5 samples. No condition was put on the percentage of gain improvement because it is too sensitive to the different split criteria.

Gain Computation Similarly to the information gain from Section 3.2.1 and the scikit-learn implementation, the improvement introduced by a split is computed as

$$\text{Gain}(S, f) = \frac{|S|}{N} \cdot \left(\text{impurity}_{node} - \frac{|S_l|}{|S|} \text{impurity}_{left} - \frac{|S_r|}{|S|} \text{impurity}_{right} \right)$$

where S is the initial set of N samples, S_l and S_r respectively are the set of samples in the left or right node and f is the considered feature.

However, the gain is only used for comparison between different thresholds and never as an absolute value. Therefore, the above formula can be simplified by removing all the constants to speed up the algorithm and becomes

$$\text{Gain}_{simplified}(S, f) = -\frac{|S_l|}{|S|} \text{impurity}_{left} - \frac{|S_r|}{|S|} \text{impurity}_{right} \,.$$

Ignored Constant Features Sometimes, some features are constant, they present the same value for all the samples in the node. Those constant features are ignored as they do not reveal any interesting piece of information. The speed of the algorithm is thus improved.

Empty Regions Split In the tree construction algorithm, the feature and the threshold showing the best gain are chosen for the split. Different thresholds are tested and it might happen that for some criteria, the impurity computation returns a constant value when the threshold locates in an empty region between two clusters.

For example, this situation will be met with the *box volume* criterion, see on Figure 4.12. Different thresholds (colored vertical lines) return the same value for the gain because the box volumes to the left and to the right of the cut are the same if the set of samples is unchanged. If the different thresholds were tested following the increasing value order, the orange line would be returned as the best threshold because no other threshold would return a strictly greater gain (they will all be equal). However, the intuition would have favoured the purple line, that splits in the middle of an empty region and not at the border of a cluster.

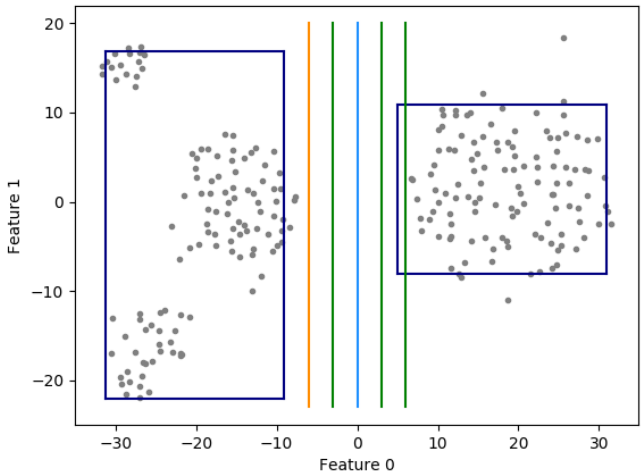


FIGURE 4.12: Different thresholds (colored vertical lines) might return the same value of the impurity criterion, as with the *box volume* criterion. In this case, the orange line will be returned as the best threshold while the intuition would have encouraged the purple line, splitting the empty regions in two equal parts. The algorithm was improved to return the middle of such plateau.

This mechanism was therefore added in the ID3 implementation to detect such *plateau* in the gain value: When the best gain is located on a plateau, the best threshold is chosen as the middle value of the plateau, the purple line from Figure 4.12.

4.4 Cluster Agglomeration

When reaching this third step, the numerous leaf nodes of the tree form subclusters. The number of clusters, or say targets, in the dataset is known. So, the subclusters will be merged to produce the expected number of bigger clusters.

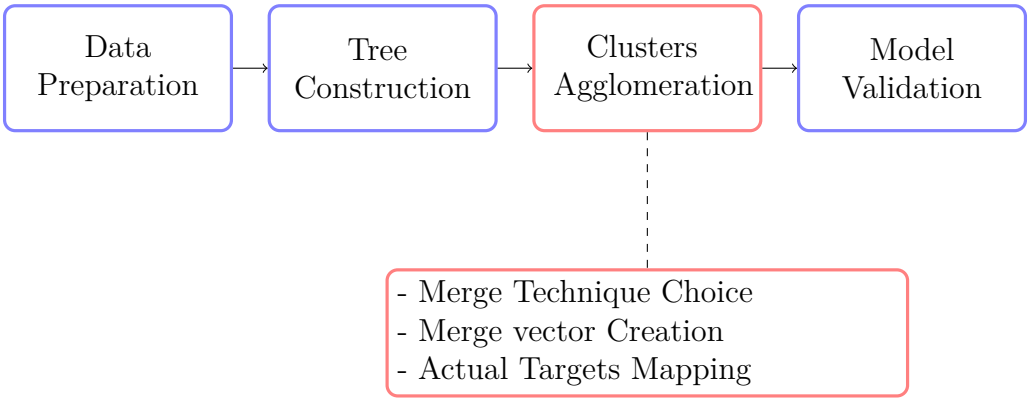


FIGURE 4.13: Algorithm steps: Clusters agglomeration

Merge Technique Choice The techniques from Section 3.3 were implemented for this thesis. The following parameters were set:

- Prototypes: For the *Set of Prototypes* method, the number of three prototypes was chosen with a α value set to 0.2;
- Box distance: The same limit of 90% was set for the number of samples in the box when growing the width;
- Graph connectivity: The k-nearest neighbours graph from the *graph distance* criterion was created with the scikit-learn library. The number of neighbours was set to 10 times the number of features. This number is increased by a factor 2 and the graph is re-created if no more link between subclusters is found.

Merge vector Creation With any merge technique, clusters from the leaf nodes are merged and assigned the same label. This is done thanks to a "merge vector" of length equal to the number of nodes in the tree. At the index of each node is given a label, a new class, in a way that the merged nodes refer to the same label.

Later on, when testing the model, samples from the test set will be guided through the tree based on their value for the feature and the threshold specific to each node. When a sample reaches the bottom of the tree, the leaf node has a certain index. The value at this index in the merge vector returns the label to assign to the sample, its prediction.

Actual Targets Mapping Luckily, the datasets used in this thesis had labels designing the actual target of each sample. It is therefore possible to verify the clustering accuracy against the ground truth. However, in this case, another mapping mechanism is needed to find the correspondence between the original and the predicted labels coming from the merge process. This mapping is done by finding the most represented original target in a cluster that consists of all samples labelled with a certain predicted class. Obviously, the training set is used for this operation as well.

4.5 Model Validation

Now that the tree is built and the leaves are merged, the time has come for the evaluation of the performance of the new split criteria and agglomeration algorithms.

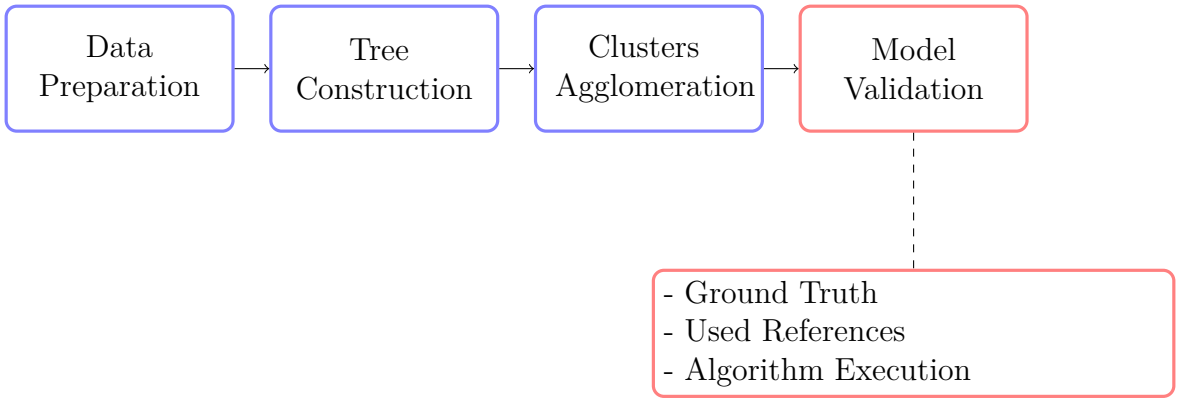


FIGURE 4.14: Algorithm steps: Model Validation

Ground Truth The datasets used for this thesis have targets associated with each sample. This target will be used as the ground truth and the misclassification rate is computed as

$$\text{Misclassification rate} = \frac{\text{number of wrongly assigned samples}}{\text{total number of samples}}.$$

This value, expressed as a percentage, indicates how well the algorithm discriminated the clusters. As low the misclassification rate, as good the algorithm at the task.

Used References The new split criteria and agglomeration algorithm will be compared against:

- A k-means algorithm directly applied on the data to form clusters without any tree;
- The supervised decision tree that uses the original targets for the split criterion and the clusters agglomeration;
- The supervised decision tree that uses the labels generated by a k-means for the split criterion and the clusters agglomeration.

For the two applications of k-means, the implementation of this algorithm in scikit-learn [17] cluster package is used. The number of centroids, and therefore of clusters, is given as input of the method: The number of targets in the original data, meaning 10 for digits and MNIST. The algorithm stops after 100 iterations.

Algorithm Execution The entry point of the algorithm is a file `main.py`. When running it, arguments from Table 4.1 are expected. Those arguments indicate what are the parameters that can vary, to generate the wanted experiments.

Name	Possible values	Note
Clustering	tree	for the re-implemented ID3 algorithm
	sklearn	for the scikit-learn algorithm
	kmeans	to use k-means, without any tree
Dataset	newsgroups	see Section 4.2.1
	digits	
	mnist	
Nb of targets	<int>	for the dataset import
Dim. reduction	tsne	see Section 4.2.2
	pca	
	None	
Nb of components	<int>	for the dimensionality reduction
Label assignment	kmeans	see Section 4.5
	None	
Max depth	<int>	to stop the tree growth
Split criterion	entropy	see Section 3.2
	boxes	
	graph	
Merge technique	targets	see Section 3.3
	single_proto	
	three_proto	
	boxes	
	graph	

TABLE 4.1: Input arguments for `main.py`

CHAPTER

5

EXPERIMENTAL RESULTS

This section shows the results of the experiments carried out in order to compare the different clustering techniques. Supervised learning is the upper bound, in terms of performance, as the knowledge of the targets gives an obvious advantage.

Organization of the experiments Experiment 1 consists in selecting the relevant datasets for the next experiments. Indeed, as explained in Section 4.2.2, either t-SNE or PCA can be used for the dimensionality reduction of *digits* and *MNIST* datasets.

This thesis contributions impact two steps of the algorithm: The tree construction with new split criteria definition, and the clusters agglomeration with the development of leaves merge mechanisms. Before performing any further tests, Experiments 2 and 3 will ensure that they make sense, that the developed algorithms give acceptable misclassification rates.

Experiment 4 compares all the combinations of split criteria and merge mechanisms, to retrieve the most efficient ones. To position the new algorithm among the existing clustering techniques, Experiment 5 involves the direct application of k-means algorithm or a possible label assignment with k-means before using supervised learning. The influence of the maximal depth is also assessed in this last experiment.

Results presentation For each experiment, 30 iterations are done. At the beginning of each new iteration, all samples are shuffled. This process ensures that misclassification rate is computed for each of the compared algorithms based on the same data. All experiments are executed for both *digits* and *MNIST* datasets, with dimensionality reduction leading to 2 or 3 dimensions data space. In the tables, $5C$ or $10C$ designates the number of classes selected from the original dataset: The selected images respectively represent figures from 0 to 5 or from 0 to 9.

In the tables, the figures are the mean and the standard deviation of the mean of misclassification rates computed for each iteration. The misclassification rate is the percentage of errors if the actual target of each sample of the test set is compared to the prediction coming from the tree (see Section 4.5).

5.1 Datasets Selection

To get meaningful results, a suitable dataset should be selected. *digits* and *MNIST* datasets require a dimensionality reduction. The first experiment is meant to define what dataset, what dimensionality reduction and how many dimensions to use in the following experiments.

The comparison is performed with 5 or 10 targets, on both datasets, in 2 or 3D, with t-SNE or PCA for the dimensionality reduction. The tree will be constructed on a supervised way, meaning that the cross-entropy is the split criterion and that the clusters agglomeration is based on the labels. By this experiment, the dimensions in which nice clusters shapes are formed are highlighted. For MNIST, for computational time reason, whatever the number of loaded targets, 2000 random samples are picked, instead of the full dataset consisting of 70 000 samples.

Table 5.1: Datasets selection: Misclassification rates.

		[% Error]	PCA 2D	PCA 3D	t-SNE 2D	t-SNE 3D
DIGITS	5 C	Mean	32.98	14.22	0.68	1.27
		Std dev	0.49	0.35	0.16	0.16
	10 C	Mean	59.1	39.31	4.4	5.73
		Std dev	0.31	0.3	0.42	0.4
MNIST	5 C	Mean	37.6	23.06	5.76	6.48
		Std dev	0.38	0.32	0.41	0.31
	10 C	Mean	67.65	56.26	22.81	26.98
		Std dev	0.42	0.5	0.76	0.55

The accuracy of the trees created after a dimensionality reduction with PCA is not sufficient, Table 5.1 shows that some classes are not discriminated against each other, giving such high misclassification rates. For this reason, t-SNE dimensionality reduction will be used in the following experiments. At first glance, digits dataset seems to form clusters easier to differentiate than MNIST, this is especially true when the 10 classes are in use.

Table 5.2: Split criteria comparison: Misclassification rates.

		[% Error]	Entropy	Box Volume	Graph Closeness
DIGITS 2D	5 C	Mean	0.79	2.75	2.07
		Std dev	0.3	0.54	0.26
	10 C	Mean	4.47	10.35	14.49
		Std dev	0.56	0.74	1.53
DIGITS 3D	5 C	Mean	1.46	2.48	6.93
		Std dev	0.16	0.23	0.98
	10 C	Mean	7.2	12.28	21.89
		Std dev	0.84	0.99	1.94
MNIST 2D	5 C	Mean	5.98	8.67	9.56
		Std dev	0.43	0.68	0.99
	10 C	Mean	22.95	32.07	33.05
		Std dev	0.81	1.17	1.41
MNIST 3D	5 C	Mean	6.37	11.41	11.89
		Std dev	0.3	0.94	0.88
	10 C	Mean	27.33	37.27	39.82
		Std dev	0.6	1.03	1.27

5.2 Split Criteria Comparison

Two split criteria are developed in the scope of this thesis: *Box Volume* and *Graph Closeness*. The purpose of this experiment is to assess whether the implementation is valid, independently of the chosen cluster agglomeration algorithm as this last one could introduce bias. The clusters agglomeration is here based on the labels. Using labels for the agglomeration does not reflect the performance of the final algorithm, but allows to find tendencies.

The experiment is run with a constant maximal depth of 5 for the tree growth. This depth is high enough, as it sometimes results in a misclassification rate of 0% with 10 targets in supervised learning. The average misclassification rate is presented in Table 5.2.

Cross-entropy, the supervised criterion, offers of course better results, it is here displayed as a reference. When coupled with a supervised merge of leaves, Box Volume seem to present better performance in finding the targets of the test set samples than Graph Closeness. Both seem valid and can be used in further implementation.

5.3 Clusters Agglomeration Comparison

Similarly to the previous experiment, to compare the clusters agglomeration without being deceived by the possible errors due to the split criterion, the different clusters agglomeration techniques will be compared when applied on the leaves nodes of a tree built with cross-entropy as split criterion. Misclassification rates for a maximal depth of 5 are shown on Table 5.3.

The two agglomeration techniques based on the prototypes seem to be more efficient in finding clusters, with similar misclassification rates except for the digits 3D where Three Prototypes performed better. Box Distance gives the higher misclassification rates.

5.4 Pairs Comparison

In order to discover what are the winning combinations among the 2 new split criteria and the 4 clusters agglomeration algorithms, they are tested in pairs in 2D and 3D with a maximum depth of 5. The results are presented in Table 5.4.

The *Graph Closeness* criterion is clearly more efficient at splitting nodes than the *Box Volume* criterion in 2 and 3 dimensions, no matter what clusters agglomeration mechanism is in use. The prototype-based techniques have again be proved to be more efficient than the two others.

For next experiment, some of the combinations are compared to other clustering techniques. From the results of Table 5.4, two prototype-based agglomeration mechanisms are chosen because they seem to merge clusters better. Still another agglomeration mechanism, Graph Connectivity, gives sometimes interesting results, so it is selected as well. In summary, the following best combinations are kept:

- *Box Volume + Single Prototype*;
- *Graph Closeness + Three Prototypes*;
- *Graph Closeness + Graph Connectivity.*

TABLE 5.3: Clusters agglomeration comparison: Misclassification rates.

		[% Error]	Based on labels	Single Prototype	Three Prototypes	Box Distance	Graph Connectivity
DIGITS 2D	5 C	Mean	0.49	5.78	7.41	16.79	15.82
		Std dev	0.12	0.95	1.23	1.67	1.62
	10 C	Mean	3.71	14.57	14.83	32.73	22.21
		Std dev	0.43	0.95	0.92	2.11	1.53
DIGITS 3D	5 C	Mean	1.51	23.32	11.69	34.97	15.16
		Std dev	0.19	2.28	1.73	3.17	2.24
	10 C	Mean	7.67	20.77	19.02	50.23	29.76
		Std dev	0.7	1.1	1.15	1.77	1.52
MNIST 2D	5 C	Mean	5.7	19.62	19.6	48.69	39.08
		Std dev	0.27	1.66	1.69	3.14	2.53
	10 C	Mean	24.39	35.57	37.29	64.73	58.8
		Std dev	0.82	0.96	1.03	1.7	1.48
MNIST 3D	5 C	Mean	6.78	29.87	29.06	61.31	54.23
		Std dev	0.51	1.99	2.38	1.49	3.17
	10 C	Mean	26.77	40.2	42.26	69.44	73.7
		Std dev	0.78	0.83	1.16	1.22	1.3

TABLE 5.4: Pairs comparison: Misclassification rates.

Agglomeration		[% Error]	Single prototype		Three prototypes		Box distance		Graph connectivity	
Split criterion			Box Vol.	Graph Cl.	Box Vol.	Graph Cl.	Box Vol.	Graph Cl.	Box Vol.	Graph Cl.
DIGITS 2D	5 C	Mean	7.04	6.18	8.75	7.28	16.35	10.77	20.75	7.41
		Std dev	0.99	0.93	1.21	0.94	2.19	1.7	1.97	1.05
	10 C	Mean	18.06	19.89	22.06	21.32	43.35	36.36	33.0	24.41
		Std dev	0.95	1.45	0.96	1.31	2.57	2.5	1.95	1.65
DIGITS 3D	5 C	Mean	11.9	8.0	13.4	7.01	38.1	17.71	27.48	11.41
		Std dev	1.61	0.98	1.73	0.89	3.8	2.96	2.23	1.61
	10 C	Mean	19.48	22.44	24.87	23.72	62.34	40.0	37.96	26.11
		Std dev	1.17	1.67	1.11	1.61	1.45	2.22	1.94	1.81
MNIST 2D	5 C	Mean	23.46	23.69	28.98	27.8	55.3	51.78	43.93	39.88
		Std dev	1.54	1.58	1.72	1.58	2.45	3.45	2.08	2.16
	10 C	Mean	38.57	39.11	41.35	40.37	66.07	59.07	56.91	53.72
		Std dev	0.95	0.83	0.82	0.73	1.18	1.2	1.23	1.27
MNIST 3D	5 C	Mean	25.8	26.69	28.52	27.24	65.25	56.89	52.27	38.83
		Std dev	1.95	1.55	1.75	1.52	0.99	1.89	2.68	2.79
	10 C	Mean	47.18	44.27	49.88	46.41	70.96	58.34	69.39	58.54
		Std dev	1.06	0.94	1.15	1.04	0.63	1.35	1.31	1.12

5.5 Other Clustering Techniques

In this experiment, the three best combinations from the previous section will be compared to other clustering techniques:

- Supervised: A decision tree is constructed with cross-entropy as split criterion and clusters agglomeration is done from the labels;
- k-means then supervised: At first, k-means algorithm is applied to detect clusters and define labels, then those labels are used to build a supervised decision tree;
- Direct k-means: Without any tree construction, k-means algorithm is directly applied.

The analysis is done for both a maximum depth of 5 in Table 5.5 and a maximum depth of 10 in Table 5.6.

Unfortunately, globally, from the results in both tables, clustering with decision trees algorithms under-perform compared to other clustering techniques, even though the misclassification rates are not exaggeratedly greater then the k-means-based algorithms. However, digits dataset reduced in 3D with 5C shows better misclassification rates than the other clustering techniques in both Table 5.5 and 5.6.

As expected, increasing the maximal depth gives lower misclassification rates. In this case, more features are used, and more leaf nodes are merged at the end.

5.6 Higher Dimensions

There is a willingness to try datasets with larger feature vectors, such as the newsgroups dataset or the digits or MNIST datasets without dimensionality reduction. However, this experiment is not executed because:

- The poor results in low dimensional space are not promising;
- The supervised learning misclassification rate lays around 20% for reasonable maximum depth of the tree for those datasets;
- The computational cost of running an unoptimized algorithm for the tree construction with high maximal depth and the clusters agglomeration is excessively high (around 60 minutes to create a single tree).

TABLE 5.5: Other clustering techniques (maximal depth = 5): Misclassification rates.

		[% Error]	Supervised	k-means then supervised	Direct k-means	Box Volume Single Proto.	Graph Clos. Three Proto.	Graph Clos. Graph Conn.
DIGITS 2D	5 C	Mean	0.69	6.31	6.15	8.49	8.17	14.13
		Std dev	0.11	0.78	0.79	1.12	0.98	1.66
	10 C	Mean	5.02	13.52	12.85	17.76	20.02	22.77
		Std dev	0.65	0.79	0.76	0.79	0.97	1.38
DIGITS 3D	5 C	Mean	1.59	16.05	15.5	13.9	11.85	14.04
		Std dev	0.2	2.03	1.79	1.53	1.35	1.54
	10 C	Mean	7.56	14.67	13.03	20.86	26.8	28.5
		Std dev	0.56	0.83	0.81	0.85	1.78	1.85
MNIST 2D	5 C	Mean	5.54	17.95	17.53	21.0	22.93	33.44
		Std dev	0.28	1.25	1.26	1.26	1.36	2.27
	10 C	Mean	8.08	11.97	11.31	13.45	13.63	18.14
		Std dev	2.22	3.14	2.99	3.49	3.55	4.65
MNIST 3D	5 C	Mean	6.22	18.44	17.25	24.14	27.41	37.34
		Std dev	0.33	1.63	1.55	1.91	1.79	3.0
	10 C	Mean	25.94	37.27	33.49	44.07	45.68	57.65
		Std dev	0.65	0.91	0.77	0.94	0.95	1.21

TABLE 5.6: Other clustering techniques (maximal depth = 10): Misclassification rates.

		[% Error]	Supervised	k-means then supervised	Direct k-means	Box Volume Single Proto.	Graph Clos. Three Proto.	Graph Clos. Graph Conn.
DIGITS 2D	5 C	Mean	0.54	5.94	5.85	8.45	7.01	8.9
		Std dev	0.13	0.68	0.68	1.18	0.99	1.31
	10 C	Mean	2.39	12.78	12.18	16.49	16.66	22.1
		Std dev	0.11	0.93	0.95	0.89	1.2	1.27
DIGITS 3D	5 C	Mean	1.38	20.41	19.59	32.77	10.06	11.99
		Std dev	0.17	2.16	2.14	2.78	1.4	1.5
	10 C	Mean	3.48	13.78	12.59	24.27	18.21	18.4
		Std dev	0.18	1.12	0.91	1.35	1.09	1.34
MNIST 2D	5 C	Mean	4.66	17.49	17.19	18.46	23.65	28.17
		Std dev	0.18	1.78	1.76	1.7	1.94	2.74
	10 C	Mean	15.3	35.38	34.5	36.35	38.89	49.16
		Std dev	0.27	0.81	0.77	0.77	0.76	1.1
MNIST 3D	5 C	Mean	5.58	19.9	18.85	31.93	31.04	37.25
		Std dev	0.27	1.68	1.65	2.27	1.33	2.15
	10 C	Mean	17.04	36.38	35.19	43.96	43.39	53.56
		Std dev	0.35	1.12	1.05	1.39	1.08	1.26

CHAPTER

6

DISCUSSION

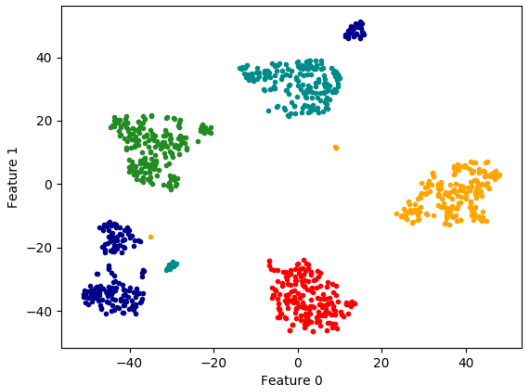
This chapter will gather thoughts and ideas on the results from the previous chapter. First by a dedicated analysis of the datasets, the split criteria and the clusters agglomeration algorithm. Secondly on the comparison with other clustering algorithms.

6.1 Dataset Selection

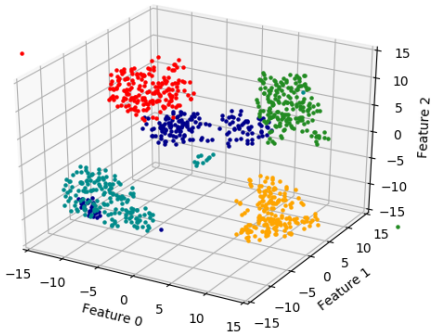
Dimensionality Reduction t-SNE algorithm tends to preserve the neighbourhood of the points: If some points are close in the high dimensional space, they will be as well in the low dimensional space. Therefore, clusters that exist in high dimensional space are maintained. On the other hand, PCA focuses on correlations in the data not on distance between samples. This is probably the reason why t-SNE forms distinct clusters while PCA does not, as reflected on the results from Table 5.1.

However, the misclassification rates from Table 5.1 are computed for supervised learning. If 10 different targets are expected but the data space is divided into 20 clusters, it might be that two clusters of the same target are not located next to each other. The agglomeration based on the labels does not mind such a situation, while the algorithms from this thesis consider that close clusters should be merged. The misclassification rate is higher in this last situation.

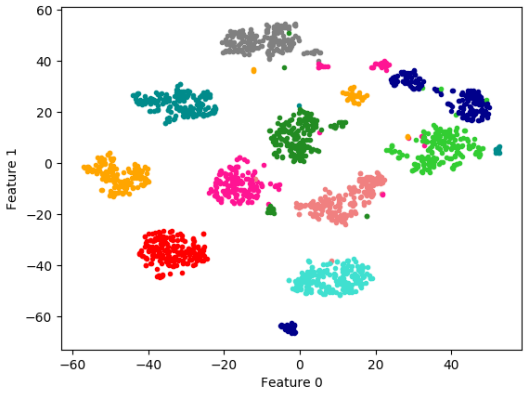
Overlapping clusters The digits dataset is the simplest in the sense that well-separated spherical clusters are present in 2 or 3 dimensions after t-SNE dimensionality reduction. This is illustrated on Figure 6.1. Clusters from MNIST dataset sometimes overlap as shown on Figure 6.2, this is the reason why the results are disappointing for MNIST in all the Tables.



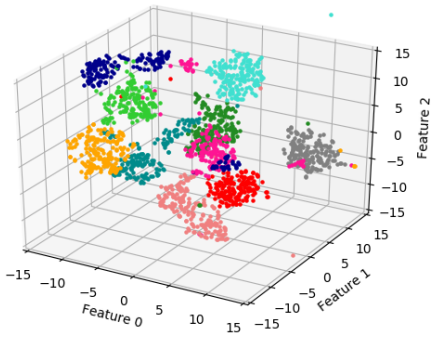
(A) 2D - 5 Classes



(B) 3D - 5 Classes

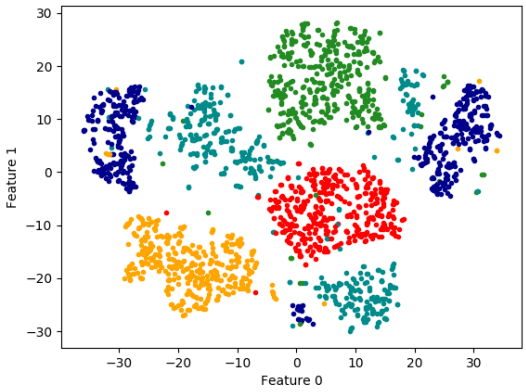


(C) 2D - 10 Classes

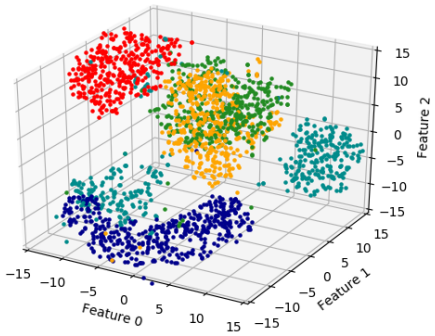


(D) 3D - 10 Classes

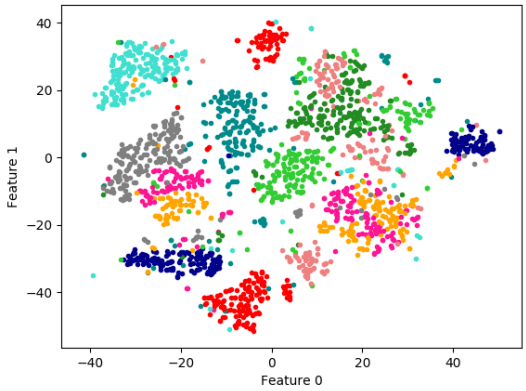
FIGURE 6.1: Digits dataset in 2 or 3D after t-SNE dimensionality reduction. The clusters in digits dataset are well-separated. The colors represent the target classes.



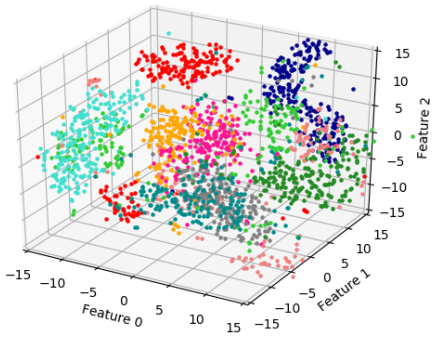
(A) 2D - 5 Classes



(B) 3D - 5 Classes

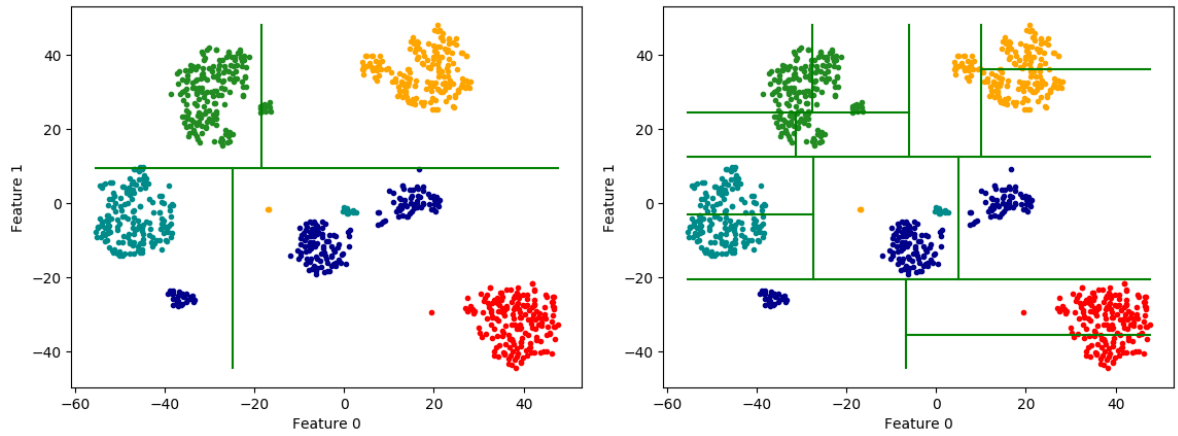


(C) 2D - 10 Classes



(D) 3D - 10 Classes

FIGURE 6.2: MNIST dataset in 2 or 3D after t-SNE dimensionality reduction. In MNIST dataset, the clusters tend to overlap each other. The colors represent the target classes.



(A) Box Volume (B) Graph Closeness

FIGURE 6.3: Comparison of the cuts generated using *box volume* or *graph closeness* for the same samples. The colors represent the target of each cluster.

6.2 Tree Construction

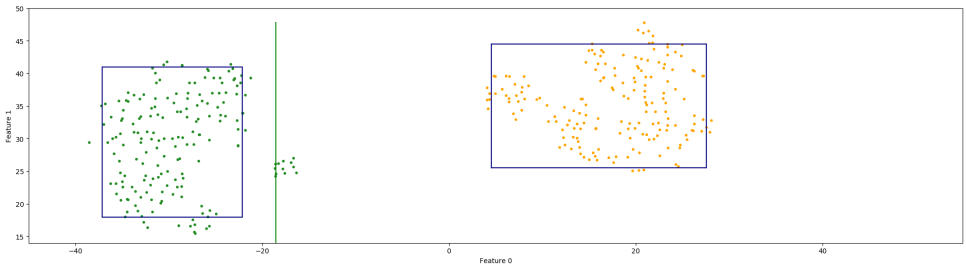
Split Criterion Accuracy Globally, the accuracies of trees constructed with the *graph closeness* split criterion are higher than those constructed with the *box volume* split criterion (see Table 5.4). As shown visually on Figure 6.3, *box volume* criterion tends to cut the data space really close to the clusters, sometimes even crop the edges, due to the 95% limit.

Surprisingly, the *box volume* criterion was performing better when the agglomeration was done with labels instead of a clustering agglomeration algorithm, as shown on Table 5.2. This can be explained by the smaller regions created when it is used for the splits. Indeed, the smaller the region, the purest. Therefore the assignment based on labels will perform better.

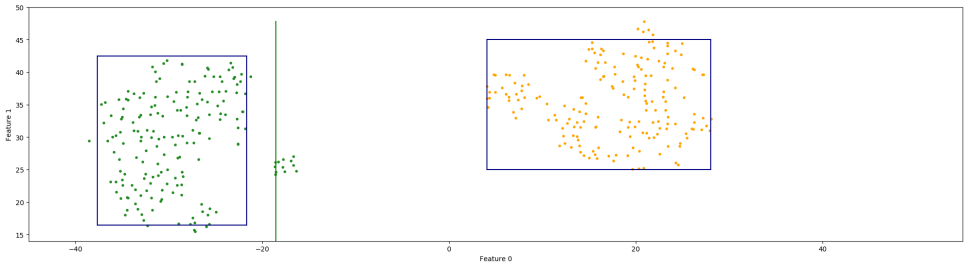
Box Volume Limit One could think that a larger limit would offer better clustering results for the box volume criterion. Actually, increasing the limit to a higher value will not help, and can additionally increase the outliers influence.

Indeed, on Figure 6.4, a little green cluster is present in between the green and the yellow clusters. Box volume split criterion tends to cut in the middle of it, in a way that the small numbers of samples from either side are not counted in the box. This behaviour is not wanted and causes a decrease in accuracy. Clusters usually have spherical or ellipsoidal shapes, so using ellipsoids instead of boxes might improve the results.

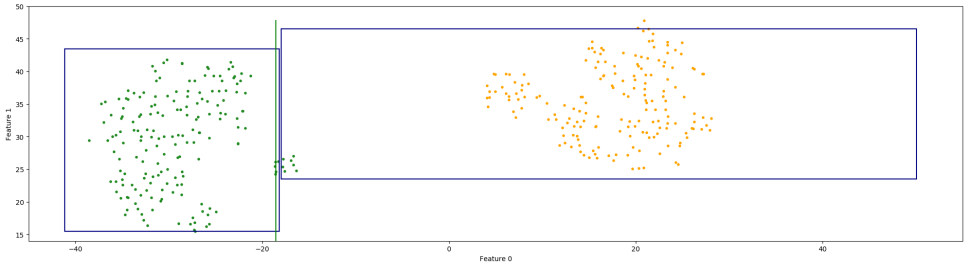
Cuts Inside Clusters On both those images, cuts are done in the middle of clusters because the tree growth does not stop until the maximum depth has been reached. So, at some point, regions containing a single cluster might be divided too much. This is not a problem as the next part of the algorithm will merge them for sure because they are located really close to each other.



(A) Limit 90%



(B) Limit 95%



(C) Limit 98%

FIGURE 6.4: Different limit for the *box volume* split criterion. When the limit is set to 98%, the box size is heavily influenced by outliers. The colors represent the target of each cluster.

6.3 Clusters Agglomeration

Accuracy Table 5.3 results shows that all clusters agglomeration algorithms implementations are valid. However, the results of this table must be challenged because there are less nodes to merge when the split criterion is cross-entropy instead of any of the two other split criteria from this thesis. Indeed, cross-entropy criterion stops splitting a node when it is "pure", i.e. nodes containing all samples for the same target. In clustering, this information is not available so a pure node will be divided regardless, and more nodes will request agglomeration at the end.

Accumulator Drawback *Box distance* gives the highest misclassification rates. This can be explained by the computation of the distance between boxes. Indeed, two boxes could overlap in one dimension but be really far apart in another. The mechanism, putting 0 in the accumulator for the first dimension, encourages the merge of those two boxes.

Prototypes According to Table 5.4, the two prototype-based agglomeration mechanisms are more efficient at merging the regions than the two others (the misclassification rates are twice smaller actually).

A single prototype approach considers that the clusters have spherical shapes and so all samples are located closer to a mean. Using several prototypes allows to spread them along the cluster and gives the possibility to fit other shapes, like elongated cluster shapes. Digits and MNIST datasets offer spherical clusters after the dimensionality reduction, so no real difference can be observed between those two clusters agglomeration techniques in Table 5.4.

Computational Cost The drawback of *Graph Connectivity* algorithm is that it is slow to compute. Indeed, after the construction of the nearest neighbours graphs, all edges should be reviewed one-by-one to determine if the edge should be kept (if the two linked samples belong to different clusters) or removed (if they are part of the same clusters). This operation is time-consuming and could be improved by keeping the information on what clusters are good candidates for the next merge, instead of re-creating each time a new graph.

6.4 Comparison with Other Clustering Techniques

Accuracy Obviously, in Tables 5.5 and 5.6, the supervised decision tree clearly surpasses the clustering algorithms, thanks to the target classes knowledge. Second are k-means-based approaches that are able to find clusters with similar misclassification rates. Unfortunately, the decision trees developed for clustering in this thesis do not hold the distance...

k-means-based Algorithms k-means followed by a supervised learning and direct k-means show similar misclassification rates, this can also be explained by the low misclassification rate of the supervised decision tree, it therefore does not make any noticeable difference if it is applied or not.

When used for the prediction, k-means algorithm requests the computation of the distance between the new sample to assign to a cluster and all the centroids. This can be time-consuming for long features vector. The interest of using k-means to label the data before feeding them to a supervised decision tree amounts to saying that the k-means is only applied for the construction of the tree. To predict the cluster of a new sample, only the tree is used and so only few features of the sample are involved.

Maximal depth Decision trees from Table 5.5 are built with a maximal depth of 5, meaning that 63 nodes, or regions, can be identified at most. Given the low dimensional space, the cuts are limited to few directions or axes. Thus, when the clusters are really close to each other, like on Figure 6.1C, it is difficult to discriminate them with so few nodes. In Experiment 5, an extra analysis was performed to check if higher maximal depth could help improve the results. This path seems promising but further investigations must be made to ensure that indeed the accuracy could someday be similar to the one achieved by k-means algorithm.

Number of iterations Direct k-means does not depend on a tree construction. For this reason, the misclassification rates from Tables 5.5 and 5.6 should not differ. The observed variations highlight that probably more iterations should be done, to be really able to compare the results. At the moment, for all the experiments, the means were computed from 30 iterations. Increasing this number to 100 is certainly a way to get more reliable results.

6.5 Higher Dimensions

Optimization By re-implementing the ID3 algorithm, as explained at Section 4.3, the computational advantage of using optimized libraries was lost. It is therefore really time-consuming to assess the behaviour of the new split criteria on data where dimensionality reduction is not applied. By working on the optimization of each part of the algorithm, running experiments on larger feature vectors could be made possible.

Maximal Depth When working with larger dimensions datasets, increasing the maximal depth of the tree is necessary. Many more features will be needed to recognize the cluster of a sample because, as explained in Section 4.2.2, 5 or 6 pixels are not enough to recognize the figure on the image. With really high maximal depths, the problem is reduced to the hierarchical clustering problem, if each node contains a single sample.

CHAPTER

7

CONCLUSION

In the scope of this thesis, an algorithm was developed to do clustering with decision trees. This algorithm is composed of 4 steps: the data preparation, the tree construction, the clusters agglomeration and the model validation.

During the tree construction, a split criterion must be chosen. The purpose of it is to highlight what feature and what threshold are the most suitable for a relevant split, node-by-node. Two new split criteria have emerged from the analysis : Box Volume and Graph Closeness. Box Volume surrounds the samples assigned to a node with a box, the impurity is the volume of this box. Graph Closeness is based on a nearest neighbours graph. By isolating a subset of edges from this graph, the impurity is computed as the inverse of the sum of edges weights.

After the tree construction, training set samples are assigned to the different leaf nodes. To keep only a known number of targets, clusters represented by the nodes need to be merged. The clusters agglomeration step is responsible for this task. Four clusters agglomeration algorithms were implemented: Single Prototype, Set of Prototypes, Box Distance and Graph Connectivity. In the two first, clusters are merged if their prototypes, i.e. representatives are the closest among all. Box Distance and Graph Connectivity correspond to the two split criteria described above, by merging the nodes surrounded by the closest boxes or linked by the higher number of edges.

Unfortunately, when the decision trees technique is compared to the existing k-means algorithm, the results are not satisfying. The misclassification rates are significantly higher, especially for the dataset with more targets or more overlapping clusters. A possible explanation is that the number of nodes is not high enough to allow the discrimination through cuts because they need to follow dimension axis. A search path for the future would be then to try higher maximal depths to see if the number of leaf nodes has an impact.

BIBLIOGRAPHY

[1] Pavel Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[2] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. *arXiv preprint cs/0011032*, 2000.

[3] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics*, 7(1):19–29, 2012.

[4] Ricardo Fraiman, Badih Ghattas, and Marcela Svarc. Interpretable clustering using unsupervised binary trees. *arXiv preprint arXiv:1103.5339*, 2011.

[5] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998.

[6] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[7] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2):107–145, 2001.

[8] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[9] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[10] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, volume 10, pages 331–339, 1995.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based

learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Moshe Lichman. UCI machine learning repository, 2013.

[13] Bing Liu, Yiyuan Xia, and Philip S Yu. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29. ACM, 2000.

[14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[15] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 1997.

[16] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

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

[18] Jan Struyf and Saso Dzeroski. Clustering trees with instance level constraints. In *ECML*, volume 7, pages 359–370. Springer, 2007.

[19] Ljupco Todorovski, Hendrik Blockeel, and Saso Dzeroski. Ranking with predictive clustering trees. In *European Conference on Machine Learning*, pages 444–455. Springer, 2002.

[20] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.

[21] Bernard Ženko. Learning predictive clustering rules. *Informatica*, 32(1), 2008.