Università degli Studi di Cagliari

**DOTTORATO DI RICERCA**

Economics and Business

Cycle XXXI

**TITOLO TESI**

# Community Detection Tree-Based Algorithm for semi-supervised clustering

Settore/i scientifico disciplinari di afferenza

SECS-S/01

| | |
|---|---|
| Presentata da: | Giulia Contu |
| Coordinatore Dottorato | Professor Andrea Melis |
| Tutor | Professor Claudio Conversano |

Esame finale anno accademico 2017 – 2018

Tesi discussa nella sessione d'esame Febbraio 2019

This thesis is dedicated to my family and my friends

# Acknowledgements

This thesis represents my journey in the research in Statistics, a difficult and still beautiful journey. I have travelled with amazing people in this three years, people that have supported me in this journey. I would like to thank many people.

The first two are Professor Mola and Professor Conversano. I have to thank them for offering me the possibility to study what I love, for believing in me when I probably could not believe in myself. I have to say thank you to Professor Mola for teaching me how beautiful Statistics can be, and the importance to understand and to analyze data carefully. And I thank Professor Conversano for supporting me during this three years: he taught me patience, which is essential to analyze data, results and methodologies with proper care. Moreover, he helped me in the preparation of all papers, all presentations and on this thesis.

The third person that has been really important in this journey is Professor Adalbert Wilhelm. He has welcomed me and treated me like one of the family during the time I spent in Jacob University.

The fourth person is Doctor Frigau, my colleague and friend, that has helped me in the study of statistics, learning R and in the realization of this thesis. Above all, I thank Luca for his friendship.

Moreover, I would like to thank Professor Gianni Bella and Professor Stefano Matta for making this trip much nicer and funnier. I want to thank them for their friendship and the time spent together.

Another person is very important for me. She is the Professor Luisa Salaris. She is a guide, a constant support and a dear friend.

I would like to thank my PhD colleagues. A special dedication is for Sara and Christelle.

Additionally, I would like to thank my brother Carlo, for supporting me in life and in this thesis. He read all the thesis and helped me improving the text quality, so thank you Carlo for everything.

The biggest acknowledgment goes to my family, that always supports me.

And last, but not least, I would like to thank my friends, my beautiful friends, for making my life better and to remember me how beautiful life can be.

# Abstract

Statistical methods improve and change in time, in line with the continuous increasing complexity of the phenomena and the size of the information available. Different approaches are combined together in order to improve the ability to analyze the data and to identify the possible relationships among them. In particular, the advantage of *supervised* and *unsupervised* learning approaches have been joined together in the last decades. But there is also a third way, which is actually a halfway: the *semi-supervised learning approach.*

This new approach has been applied on different methodologies, as for instance the cluster analysis. In fact, a variant of the traditional clustering paradigms has been proposed in order to obtain a better partitioning of the data, that considers and incorporates background knowledge. Different kinds of semi-supervised clusters have been identified in recent studies. Bair (2013) has classified them in three approaches: *partial labeled data*, *cluster with constraints* and *cluster associated with an outcome variable.* The last category is also the less developed in literature, and is generally used only in the medical domain.

The aim of this thesis is to define a semi-supervised clustering method able to identify clusters that are similar with respect to a specific outcome variable. A new algorithm will be proposed, based on the combination of two different methodologies: the tree-based method and the community detection in networks.

This algorithm is called *Community Detection Tree-Based Algorithm for Semi-supervised Clustering* (CTSC) and aims to define clusters that differ for the value of the response variable, and whose elements are similar for the same values of the outcome variable.

Three phases compose the CTSC. The pre-training and training phases focus on the application of the recursive partioning and on the definition of the proximity matrix. The matrix is pivotal to reveal the relationships between the observations, taking into account the outcome variable. Finally, the last phase, i.e. the cluster phase, focuses on the application of a community detection algorithm on the proximity matrix.

An innovative element is introduced in the algorithm: the clustering problem is transformed into a community detection problem. In fact, the cluster analysis is realized through the community detection algorithm, following the statement of Arruda et al. (2012), de Oliveira et al. (2008), Granell et al (2011, 2012). Moreover, different combinations of trees and community detection algorithms will be studied to offer a useful tool to the study of diverse phenomena and datasets. In fact, CTSC can be applied on different research areas and on different datasets.
Generally, the thesis presents some innovative aspects. Firstly, it is defined a new semisupervised cluster algorithm able to identify clusters similar respect to a specific outcome variable. Secondly, the clusters are identified combing different methodologies: tree based methods and the community detection algorithm. Thirdly, the community detection algorithms are used in the identification of clusters instead of the cluster traditional methodologies.
This thesis is composed by four chapters. The first chapter is focused on the description of semi-supervised learning, to better define a framework for the present proposal. Different algorithms will be analyzed to better comprehend their peculiarities, their structure and the goals.
The second chapter is focused on the review of the literature related to the tree-based methods, in order to evaluate the different algorithms and the splitting criteria. The second chapter aims to study the tree methodology in order to identify the most useful algorithm for the classification of the observations, taking into account a specific outcome variable.
The third chapter is focused on the study of the community detection methods. Specifically, several researches on networks and complex networks will be analyzed. Particular attention will be dedicated to the study of the different methodologies for the identification of the communities inside the networks. The aim of the third chapter is to identify the most useful algorithms to define the clusters inside the datasets.
The fourth chapter is focused on the presentation of the new algorithm. The three steps of CTSC will be defined, in order to explain how they work and their intermediate and final results. The application of CTSC on simulated data and on three different datasets is presented, with the aim to evaluate the results, the peculiarities, the limitations and the advantages of this algorithm. Then, a comparison between the CTSC and the traditional cluster algorithms will be

proposed, in order to evaluate the differences.

Finally, the possible future developments of CTSC will be presented.

# Contents

# List of Figures

# Chapter 1

# Semi-supervised Clustering

## 1.1 Abstract

The term *Statistical learning* defines a varied and wide set of tools useful to understand data. Inside this set, it is possible to identify different methodologies, that have been theorized through time. Among the oldest methodologies, it is possible to find for instance the *least squares method*, proposed by Legendre (1805) and Gauss (1809) at the beginning of the nineteenth century; the *logistic regression* theorized by Pearl and Read in 1920; the *probit model* proposed by Gaddum(1933) and Bliss (1934); and the *linear discriminant analysis* (LDA) in 1936 [181, 136, 201, 275, 272, 1, 157, 88, 122, 123, 287, 186]. In the definition of statistical methods, the most important changes took place when the computing technology allowed to improve the possibility of computation: an example of this is the *classification and regression trees* (CART) introduced by Breiman, Friedman, Olshen and Stone in the 1980s [61, 25, 203, 276, 274, 157]. The Statistical learning models can be assembled in two main groups: linear and nonlinear models. Specifically, it is possible to find the *generalized linear models* (GLM) that join together different methods such as linear and logistic regression as theorized by Nelder and Wedderburn in the 1970s [215, 189, 157]; and the *generalized additive models* (GAM) that are defined as *a flexible method for identifying nonlinear covariate effects in exponential family models and other likelihood-based regression models* [148, p. 308].
Generally, the statistical learning methods can be classified in different ways. Firstly, it is possible to distinguish between parametric and non-parametric methods. In the first case, the analysis starts with the definition of the form of function. On the contrary, in the second case it is not possible to make assumptions about the form of function, the space distribution and the classifier structure [254].
Secondly, it is possible to distinguish between *supervised* and *unsupervised* methods. The former defines models able to predict and to estimate a function and the results are deter-

mined by one or more inputs [157]. Supervised models can also be defined as a *methods that attempt to discover relationship between the input attributes and the target attribute* [254, p. 476]. Among supervised models is also possible to distinguish between *Classification Models* and *Regression Models.* The former is related to qualitative variables, the latter to numerical variables [101, 254]. Otherwise, the unsupervised methods allows to comprehend and to analyze the relationships and the structure of data [157]. In this case, it is not possible to have information about the function that is related to the data. The aim is to find the regularities in the input [10].
This chapter is focused on the study of the middle way between supervised and unsupervised learning. This new approach is called *semisupervised learning.* Particular attention will be applied on the study of the applications and the characteristics of this new form of learning. Later, the analysis will be especially focused on the application of semisupervised learning and cluster analysis. Traditional clustering approaches and the new semisupervised approach will be specifically analyzed. The aim of this chapter is to better define the framework that lies beneath this study.

## 1.2 Unsupervised, supervised and semi supervised learning

The learning of data operates on a dataset where it is possible to identify features and, in some cases, a qualitative or quantitative outcome variable [131]. Mainly, the aims can be two: to define a predictive model able to understand and predict the relationship between outcome variable and features; or to describe the organizational structure of the data. The learning is called supervised in the former case, and unsupervised in the latter. Specifically, given a labelled set of pairs of input and output $D(x_i, y_i)$, the supervised approach has the aim to learn a mapping from $x$ to $y$ , where [211, 74]:

- $y_i$ are called *labels* or *targets* of $x_i$;
- $D$ is the training set;
- $N$ is the number of training examples.

The supervised learning problems can be casted into two different areas: classification and regression. The former is used when a qualitative variable is predicted; the latter for quantitative variables [131].
The unsupervised learning operates on a training sample with $n$ instances $\{x_i\}_{i=1}^{n}$. Generally, the instances are independently and identically distributed. The aim is to reveal intrinsic structures that are embedded within the data relationships. In this case, the learning process does not consider prior information about the data, but it is guided by the

data and it tries to discover a specific and interesting structure in the data [269, 211]. Kevin (2016) has stated that the unsupervised learning can represent the typical learning of humans and animals. Moreover, he has argued that this kind of approach is more applicable than supervised learning, because it does not require the knowledge of human experts to analyze the data manually.

One of the most traditional unsupervised method is cluster analysis. It has the goal to separate the $n$ instances into groups. Other methods are, for instance, the *novelty detection*, which has the aim do identify the instances that are very different from the majority, and the *dimensionality reduction*, which has the goal to identify the lower dimensional features vector able to represent each instance, maintaining key characteristics of the training sample [330].

Finally, a new approach has emerged lately. It is called *the semisupervised learning.* It combines the strengths of the supervised and unsupervised learning paradigms. In other words, it *is somewhere between unsupervised and supervised learning* [330, p. 9]. It is located *halfway between supervised and unsupervised learning* [74, p. 17]. It attempts to include additional information to extend both unsupervised and supervised learning [330]. It operates considering both labeled and unlabeled data. For instance, the application of semi supervised learning to supervised classification has the aim to identify a better classifier $f(.)$ considering both the labeled and unlabeled data, rather than to consider only the labeled ones. The application on unsupervised clusters allows to obtain better clustering than the clustering from unlabeled data alone.

Generally, given a data set $X = (x_1, x_2, \ldots, x_{L+U})$ divided in two dataset $X_L = (x_1, x_2, \ldots, x_L)$ and $X_U = (x_{L+1}, x_{L+2}, \ldots, x_{L+U})$ where $L$ and $U$ are the labeled and unlabeled data items, the aim is to propagate labels from labeled instances to unlabeled instances in accordance to some diffusion rule.

In the implementation of semi-supervised learning methods, three different assumptions will be considered [330]:

- the first is the *Cluster assumption.* It establishes that the data points, which are located in the same high-density region, will be likely in the same class;
- the second assumption is the *Smoothness assumption.* It states that data points are probable candidates to be members of the same class, if they are near in the attribute space;
- finally, the third assumption is called *Manifold assumption.* It establishes that a set of data points that are located in a high dimensional space can be reduced to a smaller space using a nonlinear mapping function.

Moreover, the semisupervised learning can be applied on *transductive learning* and on *inductive learning* [74, p. 453]. In fact, the study of semi supervised learning was introduced by Vapnik in the 1970s. He has focused his attention on the problem of *transductive learning* [74]. The transductive learning attempts to realize the inference on the correct labels of the unlabeled data set. Its aim is to perform predictions only learning from the test points. On the contrary, the inductive learning attempts to estimate a mapping from the available data to the output variable. In this second case, the goal is to output a prediction function which is defined on the entire space $X$.
One of the applications of semisupervised learning is cluster analysis. Before to focus on semi-supervised clustering approach, in the following paragraphs will be introduced the traditional cluster methods and its characteristics.

## 1.3 Clusters analysis

Cluster Analysis is an unsupervised methodology that identifies interrelationships among data to make an assessment [156, 289, 184]. Specifically, the term *clustering* is used to identify methods to group unlabeled data [156]. The aim is to divide data into groups where the objects within the same group are very similar to each other and different from objects in other groups [326, 187, 200, 144, 242]. In other words, it attempts to partition a data set into homogeneous subgroups and can be applied to perform exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification [156, 326, 187, 144].
Clustering evolves in different steps as shown in Figure 1.1.

Specifically, It is possible to identify the following phases [156]:

- *pattern representation*, it is referred to specific elements: the cluster algorithms, as for instance the number of classes, the number of available patterns, the type and scale of the features available. Moreover, this step is composed of two activities: the *feature selection*, where the most effective subset of the original features is identified and used in clustering; and the *feature extraction*, where one or more transformations of the input features are later used to produce new salient features;

- the definition and choice of the *pattern proximity*. It is measured through a distance function defined on pairs of patterns. An example of measure is the Euclidean distance;

- *clustering*, or grouping. Different kinds of algorithms are used to cluster, as for instance the Hierarchical Clustering Algorithms and Partitional Clustering Algorithms. The former group the data into a hierarchical tree-like structure using bottom-up or

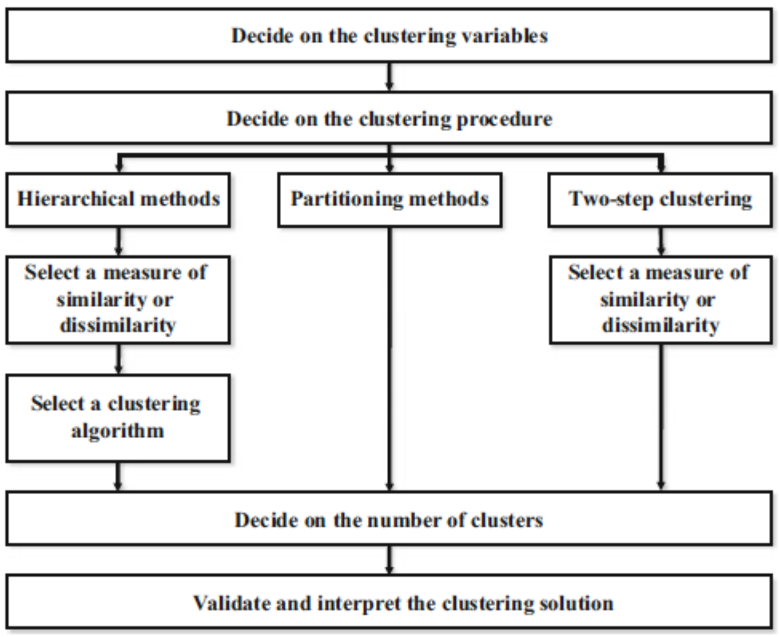

Decide on the clustering variables
Decide on the clustering procedure
Hierarchical methods
Partitioning methods
Two-step clustering
Select a measure of similarity or dissimilarity
Select a measure of similarity or dissimilarity
Select a clustering algorithm
Decide on the number of clusters
Validate and interpret the clustering solution

Figure 1.1: **The different phase in the cluster analysis, source: Sarstedt and Mooi, 2014**

top-down approaches. The main feature is that each cluster is derived from those obtained in the immediate previous stage. The latter decompose the dataset into a number of disjoint clusters that are usually optimal in terms of some predefined objective functions. Its partitions are realized considering the features of resulting clustering;

- *data abstraction*, it is a process to extract a simple and compact representation of a data set.

It is necessary to consider specific aspects while choosing the algorithm. In particular, it is possible to choose different approaches with delineated characteristics able to support the identification of clusters. Firstly, the approaches can be *agglomerative* or *divisive*. These approaches can be distinguished considering the pattern condition at the starting point. Specifically, in the first case each pattern is a distinct cluster and, later, it is merged with the other pattern until a stopping criterion is satisfied. Step by step, from the bottom to the top level the observation first, and the cluster later, are joined together. In the second case, all patterns are included in a single cluster that is then split until a stopping criterion is satisfied.
Secondly, the approach can be *monothetic* or *polythetic*. In the first case, the splitting is realized considering only one feature to divide the collection of patterns. In the second case all features are considered in the computation of distances between patterns.

Thirdly, the approach can be *hard* or *fuzzy*. The former is characterized by the fact that each pattern is allocated in a single cluster, the latter by the fact that a degree of membership is defined for each pattern and for each cluster. It must be underlined that the fuzzy clustering becomes hard when the pattern is allocated considering the largest measure of membership.

Finally, the approach can be *deterministic* or *stochastic*. Specifically, partitional approaches can use traditional techniques or random searches to optimize a squared error function.

The over mentioned characteristics are fundamental to identify the different models defined in literature. As evidenced before, the models can be casted in two main groups. The first is called *partitive clustering methods*. These algorithms identify a partition of a database $D$ of $n$ objects into a set of $k$ clusters, where the value of $k$ is an input parameter for these algorithms. Two requirements are necessary in the procedure. Firstly, each cluster must contain at least one object and each object must belong to exactly one cluster. The number of clusters is previously fixed. The algorithm evolves with an initial partition. Later, at each iteration a tentative partition is constructed by relocating the data points to optimize a conditional criterion. This procedure ends only when convergence or stability of partition occurs. Commonly partitive clustering approaches include different $k$-means type of methods such as: $k$-means, $k$-modes and $k$-median [242, 119, 119].

The most common traditional partitive cluster method is the $k$-means [23, 2]. It is usually applied on datasets with quantitative variables. The algorithm is characterized by a parameter $k$ that represents the number of clusters and a center $m_j$. Each point is assigned to the cluster whose center is nearest. Moreover, the squared Euclidean distance is used to measure the distance between observations. It is calculated as:

$$d(x_i, x_i^{'}) = \sum_{j=1}^{p} (x_i - x_i^{'})^2$$

where $x_i$ and $x_i^{'}$ are the observations from a data set with $p$ features and $x_{ij}$ corresponds to the value of the $jth$ feature for observation $i$. The aim of the k-means clustering is to assign each observation to a cluster in order to minimize the function called *within-cluster sum of squares* (WCSS). It is calculated through the formula:

$$\sum_{k=1}^{K} \sum_{C_i=k} \sum_{C_i^{'}=k} \sum_{j=1}^{p} (x_{ij} - x_{i^{'}j})^2$$

where $K$ is equal to the number of clusters, $C_i$ defines the cluster to which observation $i$ is assigned and where $1 \leq C_i \leq K$. The WCSS can be also calculated as:

$$\sum_{k=1}^{K} n_k \sum_{C_i=k} \sum_{j=1}^{p} (x_{ij} - \bar{x_{i^{'}j}})^2$$

where $n_k$ is the number of observations in cluster $k$ and $\bar{x_{ij}}$ is the mean of feature $j$ in cluster $k$.
Specifically, it involves in the following steps:

- each observation is randomly assigned to an initial cluster;
- it is necessary to calculate the mean of feature $j$ in cluster $k$ for each feature $j$ and cluster $k$;
- each observation $i$ is allocated to a new cluster $C_i$ following the formula: $C_i = argmin_k \sum_{j=1}^{p}(x_{ij} - \bar{x_{kj}})^2$;
- the steps 2 and 3 are iterated until the algorithm converges.

The characterizing element of this algorithm is the necessity to define chiefly the number of desired clusters $K$.
The second algorithm is the *hierarchical clustering.* The aim is to use distances between data groups for merging or splitting certain clusters at each step of the procedure. The hierarchical clustering can be agglomerative or divisive. Each node forms a singleton cluster at the bottom or top level of the three and later the nodes are joined together. The results of hierarchical clustering can be represented with a tree. All nodes of the tree represent a cluster. The tree is called *dendrogram.*
In literature, it is possible to identify different variants of hierarchical method and different ways to calculate the dissimilarities between clusters. Generally, the dissimilarity between cluster can be defined as $d(a, b)$, where $a$ and $b$ are the pairwise of clusters. The dissimilarities, called also linkages, used in hierarchical method are: *single-linkage*, the distance between two clusters is defined as the minimum of the distances between all pairs of patterns drawn from the two clusters; *complete linkage*, distance is defined as the maximum of all pairwise distances between patterns in the two clusters. Specifically, given two clusters $C_1$ and $C_2$, where $i \in C_1$, if $x_i$ is included in cluster $C_1$, the dissimilarity between data points $x_i$ and $x_{i'}$ is calculated for the single-linkage algorithm as [23]:

$$d(C_1, C_2) = min_{i \in C_1, i' \in C_2} d(x_i, x_i^{'})$$

and the complete linkage dissimilarity is defined as:

$$d(C_1, C_2) = max_{i \in C_1, i' \in C_2} d(x_i, x_i^{'}) = \frac{1}{n_1 n_2} \sum_{i \in C_1} \sum_{i' \in C_2} d(x_i, x_i^{'})$$

Moreover, it is possible to identify two more dissimilarities in literature. The first is *centroid linkage.* The distance between $a$ and $b$ is calculated considering centroids of the

points. Specifically, defined $x_{ij}$ as the $j-th$ component of the $i-th$ vector in $X$, it is possible to define the centroid of $X$ as the average value of $x_{ij}$, ranging over values of $i$. The second is the *average linkage*. The distance between $X$ and $Y$ is the average distance between pairs $a \in A$ and $b \in B$. Hierarchical clustering has the advantage of simplicity and the disadvantage to be heuristic [2].
Recently, researchers have observed some drawback of clustering. For instance, You et al. (2016) have underlined that the traditional cluster approaches have three limitations. Firstly, the experts do not consider prior knowledge on the dataset. Secondly, results obtained handling high dimensional data are not relevant. Finally, Bair (2013) has evidenced also that traditional supervised classification methods are not useful when it is necessary to analyze a subset of the labelled data.

## 1.4 Semisupervised clustering

In literature, it is possible to find new algorithms that attempt to overcome some of the cluster problems. Researchers have proposed a number of algorithms in order to enhance clustering quality by employing supervision approches and to solve the above mentioned problems of unsupervised cluster [39, 174]. Specifically, researchers have focused their attention on the identification of a new class of models called *semi-supervised methods.* This class has recently become a topic of significant interest [153, 326, 38, 206, 285].
The semi supervised clustering is a variant of the traditional clustering paradigms. Its aim is to obtain a better partitioning of data considering and incorporating background knowledge [132, 146, 257]. It offers the opportunity to handle and to steer the clustering process and it gives a way to interact and to manage with data to better understand them [86].
In semi-supervised clustering, two different kinds of prior knowledge are considered: *label information* and *pairwise constraints*, also called *instance-level constraints*[73, 329, 285, 140]. Givoni and Frey (2009) have stated that it is possible to identify some differences between partial labels, when a small subset of instances present labels, and instance-level constraints, when information is given in terms of constraints on pairs of data points. Firstly, the labeled data can always be used to construct instance-level constraints while the inverse does not hold in general. Secondly, the instance-level constraints do not directly provide information about the total number of clusters or classes in the data. Generally, the pairwise constraints are casted in *must-links* and *cannot-links.* The two constraints identify two points belonging to the same cluster or to different clusters [167].
Moreover, the semi-supervised clustering models can be casted in three general categories of methods: *constraint-based methods*, *distance based methods* and *hybrid methods* [228, 308, 319, 285, 322]. The aim of constraint-based methods is to handle the clustering process

with pairwise instance constraints or initialize cluster centroids by labeled instances. They use a specific constraint to guide the algorithm to obtain the appropriate data partitioning [39]. Finally, the algorithm modifies the objective function in order to respect the provided constraints [228].

Instead, the distance based approches uses the constraints in order to learn a new distance metric and group instances. Specifically, it is characterized by the definition of a clustering distortion measure able to define a good partition.

Lastly, the hybrid approach combines the two above-mentioned approaches under a probabilistic framework.

In addition, the semi supervised methods are casted with respect to the similarity measure and the pairwise constraints. Specifically, Grira et al. (2005) have classified the methods in similarity-adapting methods, where the similarity measures are adapted in order to respect the available constraints, and in search-based methods, where the clustering algorithm is modified in order to consider the constraints and the labels to better perform clustering. The use of constraints can modify the results of clustering algorithms. Finally, an interesting classification of semisupervised clustering was proposed by Bair (2013). He has defined the clustering algorithms basing on the nature of the known outcome data. Specifically, he has identified three approaches: the partially labelled data; the cluster with constraints; and the cluster associated with an outcome variable.

### 1.4.1 Partially labeled data

The clustering algorithms known as *partially labelled data* are based on label information. The cluster assignment for a subset of the data is known previously. In other words, the classification of some observations is known before starting the clustering process. Consequently, the cluster analysis involves classifying the remaining unlabeled observation considering the known cluster assignment for labeled data. Labelled and unlabelled data are used in the algorithm [46, 282, 46]. The goal is to understand both how combining labeled and unlabeled data can change the learning behavior, and how to design algorithms to take advantage of such a combination [330].

Inside this group, it is possible to identify different models. For instance, Basu et al (2002) have proposed two different models. The first is a generalization of k-means clustering. It is useful to cluster data where class labels are known for a subset of observations. To identify the clusters, they have defined $x_i$ and $x_i^{'}$ as the observations from a data set with $p$ features, and $x_{ij}$ as the value of the $jth$ feature for observation $i$. Moreover, they have supposed the existence of a subset $S_1, \ldots, S_K$ of the $x_i^{'}$ such that $x_i \in S_k$. Finally, they have defined the algorithm with different steps as shown in the Box 1.1

1. For each feature $j$ and cluster $k$, the initial cluster means is calculated as:

$$\bar{x}_{kj} = \frac{1}{\|S_k\|} \sum_{i \in S_k} x_{ij}$$

2. Each observation $i$ is located to a new cluster $C_i$. If $x_i \in S$, defined $C_i = S_k$ then $x_i \in S_k$. Otherwise we consider:

$$C_i = argmin_k \sum_{j=1}^{p} (x_{ij} - \bar{x_{kj}})^2$$

3. For each feature $j$ and cluster $k$, it is necessary to calculate the mean of feature $j$ in cluster $k$ $\bar{x_{kj}}$;
4. It is necessary to repeat steps 2 and 3 until the algorithm converges.

Box 1.1: Generalization of k-means clustering

The second model proposed by Basu et al. (2002) is called *seeded k-means clustering.* It differs from the traditional k-means clustering from the first step in the procedure. Specifically, it identifies a subset called *seeded set* $S \subseteq X$. For each $x_i \in S$, it is provided the partition to which it belongs. The partition of the *seeded set* $S$ represents the *seeded clustering* and it is used to guide the k-means clustering [38]. The algorithm involves different phases as shown in box 1.2.

Set of data point $X = \ x_1, \ldots, x_N\ $, $x_i \in R^d$, number of cluster $k$, set $S = \cup_{l=1}^{K} S_i$ of initial seeds.
Disjoint $K$ partitioning $\ X_l \ {}_{l=1}^{K}$ of $X$ such that KMeans objective function is optimized.
1. Initialize: $\mu_h^{(0)} = \frac{1}{\|S_h\|} \sum_{x \in S_k} x_i$, for $h = 1, \ldots, K$, $t \leftarrow 0$
2. Must repeat until convergence
2a. Assign each data point $x$ to the cluster $h^*$ for $h^* = argmin\|x - \mu_h^{(t)}\|^2$

Box 1.2: Algorithm Seed-Kmeans

Another interesting algorithm has been proposed by Shental et al. (2003). They have defined a framework for semi-supervised clustering that considers information in form of *equivalence constraints.* It is called *generalized Expectation Maximization algorithm* (EM algorithm). In this model, the equivalence constraints are defined as *binary functions of pairs of points, indicating whether the two points come from the same source or from two*

*different sources* [267, p. 466]. If the points derive from the same source, they are called *positive constraints*, otherwise they are called *negative constraints.* The former are handled through the EM procedure, the latter with the Generalized EM procedure using a Markov network.
To sum up, these methods combine labeled and unlabeled data with the aim to use previous information that can improve the learning approach and the similarity of partition.

### 1.4.2 Clusters with constraints

The clusters with constraints are based on the presence of complex relationships among the observations. Specifically, in this case it is possible to previously identify and recognize specific relationships among the observations. Two kinds of constraints have been defined in literature: *Must-Link Constraints* and *Cannot-Link Constraints.*
Specifically, defined $S = \{s_1, s_2, \ldots, s_n\}$ as the set of points which must be partitioned into $K$ clusters, $s_i$ and $s_j$ as any pair of points in $S$ and $d(s_i, s_j)$ as their distance, it is possible to define the *Must-Link Constraints* as a constraint that involves joining together in the same cluster a pair of point $s_i$ and $s_j$ with $i \neq j$ [95, 193, 20, 46, 309]. On the contrary, it is possible to identify the *Cannot-Link Constraints* as the constraint characterized by the fact that the pair of distinct $s_i$ and $s_j$ are allocated in different clusters under a general probabilistic framework [39]. These types of constraints are useful to define groups composed by similar instances. Moreover, they support the encoding background knowledge even when class labels are not known a priori [169]. Finally, the consideration of constraints supports the clustering algorithm to define more accurately the clusters of this partition.
Inside the constraint methods, it is possible to identify different models. For instance, a model has been proposed by Wagstaff and Cardie (2000). They called it the *COP-COBWEB.* It is based on the COBWEB method, that was theorized by Fisher in 1987, which consists in an incremental clustering algorithm [292, 180]. COBWEB considers the concept of category utility to define a clustering that maximizes inter-cluster dissimilarity and intra-cluster similarity. It refers to four operators: *add, new, merge, and split* to incorporate a new instance into the top level of the existing hierarchy. It involves applying the operator that maximizes the category utility of the resulting hierarchy. Wagstaff and Cardie (2000) have modified COBWEB introducing the concept of constraint. Specifically, they have defined $D$ as dataset, $Con_=$ as set of must-link constraints and $Con_{\neq}$ as a set of cannot-link constraints. They have established that the algorithm considers each instance $D_i$ in the dataset. If there is some must-link constraint that establishes that $D_i$ is in the same cluster as $D_j$, it is reinforced the constraint that includes $D_i$ in cluster $C_j$. Otherwise, in the algorithm the *add*, *new*, and *merge operators* are used to establish where to allocate

$D_i$. Specifically, to add an instance to a specific cluster $C_j$, it is necessary to evaluate the existence of a cannot-link constraint that would prevent $D_i$ from joining $C_j$ [292]. Later, it is necessary to create a new singleton cluster for $D_i$. However, it is possible to merge two clusters only checking the presence of any cannot-link constraints that would invalidate the merge. At the end, the partition that presents the highest value of category utility is chosen.

Wagstaff et al. (2001) have proposed a general method able to consider and incorporate background knowledge in the form of instance-level constraint into the k-means clustering algorithm. It is a modification of K-means clustering and is called *COP- KMEANS* [319, 293]. The algorithm involves different steps, as shown in the Box 1.3. Inside the algorithm, both must-link and the cannot link constraints are considered, in order to generate a partition that satisfies all given constraints.

```
COP-KMEANS data set $D$ must-link constraints $Con_= \subseteq DxD$, Cannot-link constraints $Con_{\neq} \subseteq DxD$
1. Let $C_1, \ldots, C_k$ be the initial cluster centers
2. For each point $d_i$ in $D$, assign it to the cluster $C_j$ such that
VIOLATE-CONTRAINTS $(d_i, C_j, Con_{\neq})$ is false. If no such cluster exists, fail $(return\{\})$
3. For each cluster, $C_i$ updates its center by averaging all of the points $d_j$ that have been assigned to it.
4. Iterate steps 2 and 3 until converge
5. Return $\{C_1, \ldots, C_k\}$
VIOLATE-CONSTRAINTS (data point $d$, cluster $C$, must-link constraints $Con_= \subseteq DxD$, Cannot-link constraints $Con_{\neq} \subseteq DxD$
1. For each $(d, d_= \in Con_=$, if $d_= \notin C$, return true.
2. For each $(d, d_{\neq} \in Con_{\neq}$, if $d_{\neq} \in C$, return true.
3. Otherwise, return false.
```

Box 1.3: Constrained K-means Algorithm

The two models, COBWEB and COP-KMEANS, differs only because in COP-KMEANS each observation is allocated to the nearest cluster such that no constraints are violated.

Davidson and Ravi (2005) have proposed a k-Means algorithm considering four different types of constraints. They have identified two different kinds of constraints that are different form the Must-Link and Cannot-Link Constraints. Specifically, they have identified *$\delta$ Constraints* or *Minimum Separation Constraints*, and *$\epsilon$ Constraints*. The *$\delta$ Constraints* identify the minimum value of the distance between any pair of points in two different clusters. Specifically, identified a pair of cluster $S_i$ and $S_j$ with $i \neq j$ and a pair of point $s_q$ and $s_p$ such that $s_q \in S_i$ and $s_p \in S_j$, the distance must respect the constraint $d(s_q, s_p) \geq \delta$. In other word, the distance between a pair of points has to be at least $\delta$. Inside the second group, the value of the constraint is fixed to $\epsilon \geq 0$. Moreover, it is established that given a $S_i$ cluster containing two or more point, the distance is equal to $d(s_q, s_p) \leq \epsilon$ [95]. Davidson and Ravi (2005) have defined different models considering the constraints in singular way or combined between them, as show in Box 1.4 and 1.5.

```
Defined a given collection $C$ of must-link constraints, it can be transformed into an
equivalent collection $M = M_1, M_2, \dots, M_r$ of constraints, by computing the transitive
closure of C.
1. Compute the transitive closure of the constraints in $C$. Let this computation result
in $r$ sets of points, defined by $M_1, M_2, \dots, M_r$
2. Let $S^{'} = S - \cup_{i=1}^{r} M_i$ ($S^{'}$ denotes the subset of points that are not involved in any
must-link constraint)
3. If $r \geq K_l$ then
(a) Let $A = (\cup_{i=K_l}^{r} M_i) \cup S^{'}$
(b) Output $M_1, M_2, \dots, M_{K_l - 1}, A$ else
$|S^{'}| < K_l - r$ then Output "There is no solution."
else
(a) Let $t = K_l - r$. Partition $S^{'}$ into $t$ clusters $A_1, A_2, \dots, A_t$ arbitrarily.
(b) Output $M_1, M_2, \dots, M_r$, $A_1 A_2, \dots, A_t$
```

Box 1.4: Algorithm for the ML-Feasibility Problem

```
Algorithm for the $\delta$ Feasibility Problem
Let $S = \{s_1, s_2, \dots, s_n\}$ denote the given set of points which must be partitioned into
$K$ clusters
1. For each point $s_i$ do
(a) Determine the set $x_i \subseteq S - S_i$ of points such that for each point $x_j \in X_i$, $d(s_i, x_j) <$
$\delta$.
(b) For each point $x_j \in X_i$, create the must-link constraint $s_i, x_j$.
2. Let C denote the set of all the must-link constraints created in Step 1. Use the
algorithm for the MLfeasibility problem with point set $S$, constraint set $C$ and the
values $K_l$ and $K_u$.
```

Box 1.5: Algorithm for the $\delta$ Feasibility Problem

The above-mentioned models refer to the k-means clustering algorithm. However, in literature it is also possible to find other models, based on the hierarchical clustering method. For instance, Miyamoto and Terami (2010) have proposed a hierarchical algorithm with pairwise constraints. They have taken into consideration the presence of must-link and cannot-link constraints and they have combined the single linkage with the Kernel function. Specifically, they have introduced the pairwise constraints in the single linkage and

have used Kernel approach to overcome the problem related to the centroid method and the Ward method.
Bade and Nurnberger (2006) have created another model, called *Biased Hierarchical Agglomerative Clustering* (BiHAC). It is a hierarchical agglomerative clustering algorithm that uses must-link-before (MLB) to supervise the clustering process.
Zhao and Qi (2010) have proposed a model that is based on a new type of constraint for hierarchical agglomerative clustering. It is called *ordering constraint* (OC). The algorithm is based on the idea that if the instance $A$ is more similar to the instance $C$ than the instance $B$, it is necessary to define a ordering constraint that establishes that $A$ must merge with $C$ before merging with $B$ during clustering. After introducing ordering constraints, the algorithm involves realizing a hierarchical agglomerative clustering. The aim is to built a dendrogram that respects the ordering constraint. Zhao and Qi (2010) have called the algorithm *HACOC*. The researchers identified three advantages of this algorithm. Firstly, it uses prior knowledge that provide to prevent many inaccurate merging operations. Secondly, it generates a stable dendrogram. Finally, it is possible to obtain different dendrograms considering various ordering constraints.
In addition, Tang et al. (2007) have introduced the *Semi-supervised Clustering method based on spheRical K-mEans via fEature projectioN* (SCREEN) (Box 1.6). They have solved the problem of constraint-guided feature projection with the semi-supervised clustering algorithms. They have included inside the model must-link and cannot-link constraints. Later, as showed in Figure 1.2 and 1.3, they have eliminated the must-link constraints and they have used the average instances $a$, $b$ and $c$ in each closure to represent the original cannot-link constraints. The size of each transitive closure becomes the weight of the representative instance.

Then, they have combined a K-means for semi-supervised clustering with constraint-guided feature projection approach ($SCREEN_{PROJ}$) to further improve the performance of semi-supervised clustering in the high-dimensional datasets [285]. Specifically, they have attempted to maximize the objective function:

$$f(x_1, x_2) = \sum_{(x_1,x_2)\in C_{CL}} \|F^T(x_1, x_2\|)^2 - \|F^T(x_1, x_2)\|^2$$

with the constraints:

$$F_i^T F_j = \begin{cases} 1 & if i = 1 \\ 1 & if i \neq 1 \end{cases}$$

where $F$ is the projection matrix whose column vectors are orthogonal to each other. The function $f$ can be also written as:

$$f(x_1, x_2) = \sum_{(x_1,x_2)\in C_{CL}} \|w_1 w_2 * F^T(x_1, x_2)\|^2$$

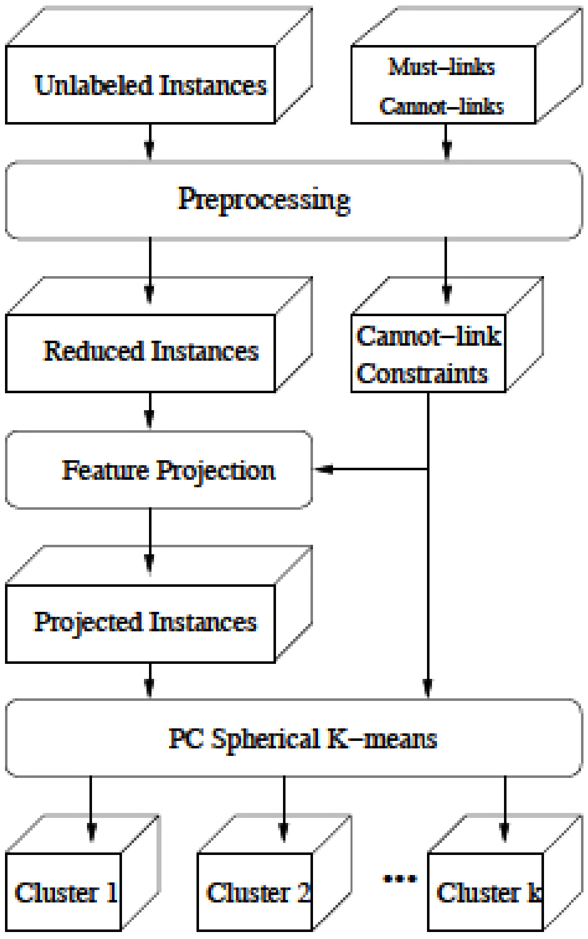


Figure 1.2: **The Framework of the SCREEN Method, source: Tang et al. (2007)**

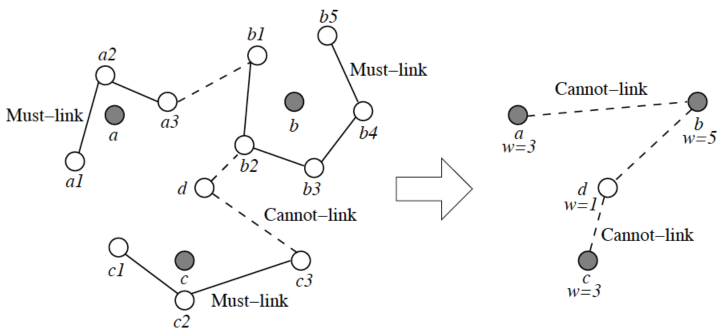


Figure 1.3: **Initialization of SCREEN Method, source: Tang et al. (2007)**

where $w_i^{'}\ _{i1^{N^{'}}}$ identifies the set of weights for the reduced instances after pre-processing to the must-link constraints and $N^{'}$ is the reduced size of instances.

```
Algorithm: SCREEN Input: Set of unit-length instances
$X^{'} = x_i^{'}\ _{i1^{N^{'}}}$, set of corresponding weight $W = w_i^{'}\ _{i1^{N^{'}}}$,
set of cannot-link constraints $CCL = \{(x_i^{'}, x_j^{'})\}$, and number of clusters $K$.
Output: $K$ partitions of the instances.
Steps:
1. $X^{'} = SCREEN_{PROJ}(X, W, C_{CL})$
2. $X_k^{'} = PCSKM(X^{'}, W, C_{CL})$, where $k = 1, \ldots, K$
3. post-process $X_k^{'}$ to be in accordance with the original instances $X$.
```

Box 1.6: Algorithm: SCREEN

### 1.4.3 Cluster associated with an outcome variable

In *Cluster associated with an outcome variable*, the clusters are realized considering a given outcome variable, and defined considering previous information of the outcome variable.
In literature, it is possible to find few methods that consider outcome variable [23].
One of the early methods has been theorized by Bair and Tibshirani (2004). The model has been developed to diagnose cancer more accurately, taking into account both the genetic profile of a tumor and the patient survival. For this reason, they have proposed a semi supervised method useful to identify subtypes of cancer that are both clinically relevant and biologically meaningful. They have analyzed $n$ patients to measure the expression level of $p$ genes for each patient, where $n \gg p$. Specifically, they have attempted to identify *a classifier that can diagnose which type of cancer a future patient will have, given the expression levels of the patient's p genes* [24, p. 513]. The algorithm involves two principal phases. Firstly, for each feature in the data set, a test statistic $T_j$ is calculated for testing the null hypothesis of no association between the *jth* feature and the outcome variable. Three different cases of outcome variable are taken into account. Specifically, if the outcome variable is [24, 171, 23]:

- binary, $T_j$ may be a t-statistic;
- continuous, $T_j$ is a t-statistic for testing the null hypothesis that the regression coefficient is equal to 0;
- a right-censored survival time, $T_j$ is the corresponding test statistic from a Cox proportional hazards model. The estimated Cox score represents a measure of the association between the gene's expression level and patient survival.

Secondly, a threshold $M$ is chosen though the cross-validation and, later, the k-means clustering is applied to the features for which $T_j > M$. The choice to use the k-means clustering is affected by the prior biological knowledge and the researchers have chosen a value of $k = 2$. Finally, the *nearest shrunken centroid* (NSC) [286] is used to assign the future patients to one of the $k$ groups previously identified, and then the *partial least squares* (PLS) [157] are used to calculate a corrected Cox Score. The aim is to identify a better measure to build the different clusters. Bair and Tibshirani (2004) have demonstrated that choosing the genes with the largest absolute corrected Cox-score produces better clusters than those obtained using the largest raw Cox scores. They have demonstrated that their method performs significantly better than the existing methods and it also has the advantage to select and use only a reduced number of genes as predictors.
Another method has been theorized by Koester et al. (2010). Specifically, they have stated that the unsupervised learning procedure is not able to identify cancer subtype correlated with the patient survival. Moreover, Koester et al. (2010) have stated that the supervised learning approach can obtain good results but with the limited biological interpretation [171].
The semi supervised method proposed by Koester et al. (2010) differs from the method of Bair and Tibshirani (2004) because Koester et al. (2010) has not used the k-means clustering method. Koester et al. (2010) have proposed a method called *semi-supervised recursively partitioned mixture model* (SS-RPMM). The aim of the model is to find cancer subtype associated with patient survival.
The algorithm involves firstly selecting a group of features that are most strongly associated with the outcome variable, and then applying the *recursive partitioned mixture model* (RPMM) proposed by Houseman et al. (2008).
Specifically, the RPMM is based on a beta distribution. Given a subject $i$, defined $Y_i = (Y_{i1}, \ldots, Y_{ij})$ as a vector of $J$ continuous outcomes, which assumes values between 0 and 1, Hoiseman et al. (2008) have assumed a distribution $f(Y_{ij} = y|C_{ij} = k; \Theta_{kj})$ where $\Theta_{kj}$ is a vector of parameters that depend on class $k$ and $j$. They have defined an algorithm that starts fitting a two-class model to the dataset. This phase allows to define two sets of weights that identify the posterior class membership in the two classes. This weight is equal to $w_{ik} = P(C_i = k|Y_i)$. The algorithm involves applying the weighted likelihood EM algorithm [102] with the aim to obtain the two classes corresponding to the new split. Finally, it can fail and the recursion is stopped, or can involve defining new weights and new recursions. The two possibilities are related to the data.
Koester et al. (2010) have stated that the RPMM model is able to estimate the number of clusters in a robust and computationally efficient manner. For this reason, they have

adapted the model and transformed it in the SemiSupervised(SS)-RPMM. This model is based on a variable $Z$, specifically $Z = (T_i, d_i)$ where $T$ is the observed failure and $d$ is an indicator that measures whether the event was observed or not. Moreover, the proportional hazards model is used through the formula:

$$h(t_i) = exp(y_0) + X_i^T \delta + \sum_{k=2}^{K} y_k I[(C_i = k h_o(t)]$$

where $X_i$ are the additionally patient information and $I(C_i = k)$ is an indicator for the class membership in the $k-th$ class for the $i-th$ patient. Moreover, they have considered and they have applied, as done before by Bair and Tibshirani (2004), the Cox model as:

$$h(t_i) = \exp(y_0) + X_i^T \delta + y_j Y_{ij}$$

in which $Y_{ij}$ represents the expression $j$ in subject $i$.
Koester et al. (2010) have stated that their model is able to combine the strengths of the semi supervised model proposed by Bair and Tibshirani (2004) and the ability of the RPMM to identify the number of clusters in a robust and efficient way. However, Gaynor and Bair (2013) have affirmed that it is possible to identify a disadvantage on the SS-RPMM. Specifically, the disadvantage is determined by the fact that some features are discarded in the first phase and they are not considered in the following phases of the algorithm. In addition, it is possible that features with a weak association with the outcome variable are used for the definition of clusters determining an uncorrected identification of the final partitions.
To overcome this problem, Gaynor and Bair (2013) have proposed another method called *supervised sparse clustering* (SSC). It is a modification of the method proposed by Witten and Tibshirani (2010), called *sparse clustering.* The SSC attempts to maximize the equation:

$$\sum_{j=1}^{p} w_j \Big( \frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} (x_{ij} - x_{ij}^{'})^2 - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n} \sum_{C_i=k}^{n} (x_{ij} - x_{ij}^{'})^2 \Big)$$

where:

- $x_{ij}$ is an observation of a dataset composed by $n$ observations and $p$ features and partitioned into $K$ cluster;
- $n_k$ is the number of observations in cluster $k$;
- $w$ is the weight attributes to each feature.

The maximization is realized through the use of an algorithm that sets the value of $w_j = \frac{1}{\sqrt{n}}$ and updates $w_j^{\prime} s$ iteratively, where $s$ is a tuning parameter used to perform the cluster.

When the value of $s$ increases, the number of $w_j's = 0$ decreases. Choosing correctly this value allows to identify clusters using only a subset of features for which $w_j > 0$. The choice of initial feature weight is realized as follows [137]: firstly, a test statistic $t_j$ is calculated, or testing the null hypothesis of no association between the $jth$ feature and the outcome variable; then, it is chosen a threshold $M$ useful to define the initial weights as:

$$w_j = \begin{cases} \frac{1}{\sqrt{m}} & \text{if} \|T_j\| & > M \\ 0 & \text{if} \|T_j\| & \leq M \end{cases}$$

where $m$ is the number of features such that $\|T_j\| > M$. Later, the value of the weight feature is updated until the procedure converges.

To sum up, these methods use the information of the outcome variable to define a better partition of data. These methods involve considering a specific dataset that is highly correlated with the outcome variable, or a specific variable able to identify similar groups.

### 1.4.4 Other methods

There are also other semisupervised clustering algorithms that have not been considered by Bair (2013). Those methods are based on distance measures and constraints with the aim to learn a new distance metric to group instances. Specifically, they are characterized by the definition of a clustering distortion measure able to define a good partition.

These algorithms involve learning firstly a distance metric from the given pairwise constraints. Then, a linear transformation is defined starting from the learned distance metric, that is later applied to generate a new vector representation for the data points. Finally, data partition is computed applying the existing clustering algorithms to the transformed vector representation [289].

Different distance measures are also defined in literature. Their definition allows evaluating the distance taking into account the different weights or the importance factors. Moreover, they can improve the performance of clustering or classification algorithms, such as K-Means and K-Nearest-Neighbor (KNN).

Other methods combine constraint-based methods and distance based methods. For instance, Bar-Hillel et al. (2003) have introduced a model that modified the Relevant Component Analysis (RCA) algorithm including constraints. Specifically, they have considered the side-information in the form of equivalence relations to learn a Mahalanobis metric. Xiang et al. (2008) have defined the Mahalanobis distance as *a measure between two data points in the space defined by relevant features* [309, p. 3]. It is possible to calculate it considering two data instances through the formula:

$$d_M(x_i, x_j) = \sqrt{(x_i, x_j)^T M (x_i, x_j)}$$

where $x_i$ and $x_j$ are two instances and $M$ is a non-negative and triangle-inequal matrix. The RCA is used to identify and down-scale global unwanted variability within the data. The model involves changing the feature space used for data representation through a global linear transformation [27]. Specifically, it assigns large weights to relevant dimensions and low weights to irrelevant dimensions. The relevant dimensions are calculated through the *chunklets* that can be defined as *a subset of points that are known to belong to the same although unknown class; chunklets are obtained from equivalence relations by applying a transitive closure* [27, p. 12]. The aim of RCA is to reduce the number of clusters. Inside the RCA, Bar-Hillel et al. (2003) have introduced an information theoretical criterion. It establishes that when an input $X$ is transformed into a new representation $Y$, it is possible to maximize the mutual information $I(X, Y)$ between $X$ and $Y$ under suitable constraints. Specifically, the researchers have attempted to obtain:

$$min_B \frac{1}{p} \sum_{j=1}^{K} \sum_{i=1}^{n_j} \mid\mid x_{ji} - m_j \mid\mid s.t. \mid B \mid \geq 1$$

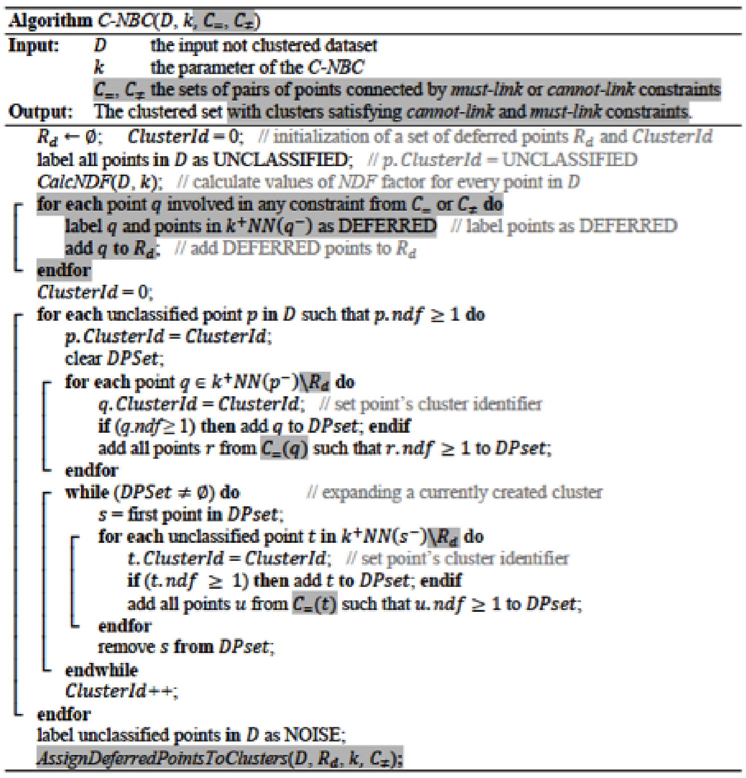
where

- $x_{ji}$ identifies a set of chunklets of data points;
- $m_j$ defines the mean of points in chunklet $j$ after the transformation;
- $P$ is the total number of points in chunklets;
- $B$ denote a Mahalanobis distance matrix.

They have defined a method that can be used considering different criteria and it supports and improves the cluster analysis.
Givan and Frey (2005) have extended the affinity propagation model considering instance-level constraints. They have considered instance-level constraints for some input data points and have incorporated constraints able to alter the similarities between data points. Specifically, they have attempted to define a model able both to combine instance-level constraints with a clustering algorithm and to find a clusters using side information. The model involves considering the maximal similarity between must-link points, the minimal similarity between cannot-link points and the shortest-path similarity between any other two points. Moreover, the "meta-points" have been introduced in the model. These points are computed to find a closure of the must link constraints. They have been added at each resulting group and at the points in a cannot-link constraint. Givan and Frey (2005) have applied their model on the analysis of the interactive image segmentation.

Zhou et al. (2005) have proposed another model called *Neighborhood-Based Clustering algorithm* (NBC). It is based on the concept of *Neighborhood*, specifically on neighborhood relationship among data. This concept is used as an object to build a neighborhood based clustering model to discover group. The main concept of NBC is the *Neighborhood Density Factor* (NDF), which is a measure of relative local density [179]. NBC is characterized by the ability to measure relative local densities. It is able to discover clusters of different local densities and of arbitrary shape. It involves two phases. The first is focused on the evaluation of NDF, the second on the clustering of dataset. Specifically, NBC evolves calculating the value of NDF of a points $p_i$ included in a database $D, i = 0, 1, \ldots, \mid D \mid$. If the value of NDF is equal to 1, $p_i$ is assigned in the cluster $c$. Later, a variable to store the dense points is introduced, called $DPSet$. It is used to cluster in an iterative process of assigning points. Specifically, the temporary variable for storing references to point, called $q$, is cleared, and the $k^+$-neighborhood of points is assigned to $c$. If $q$ is equal to 1, this point is added to $DPSet$, otherwise it is omitted and the next point is analyzed. The algorithm is repeated until no point in $D$ presents an NDF value equal to 1.
It is possible to identify other variation of NBC. For instance, Lasek (2014) has combined the known NBC algorithm and instance-level constraints such as must-link and cannot-link. The model is called *C-NBC algorithm.* Inside the model, deferred points are considered. Lasek (2014) has stated that *a point p is called deferred if it is involved in a cannot-link relationship with any other point or it belongs to a $k^+_-$-punctured neighborhood $k^*NN(q^-)$, where q is any point involved in a cannot-link relationship* [179, p. 117]. The algorithm evolves, as shown in Figure 1.4, in two parts. Firstly, the deferred points are not assigned to any cluster and these points are used to calculate NDF factors. Secondly, deferred points are allocated to appropriate clusters. C-NBC allocates deferred points to appropriate clusters. The allocation of clusters is realized considering the *Assign Deferred Points To Clusters function* (Figure 1.5). It assigns points to clusters after checking if such an assignment is feasible.
The difference between C-NBC and NBC is that the former introduce deferred points inside the model and it attempts to find a deal with must-link constraints when building clusters.

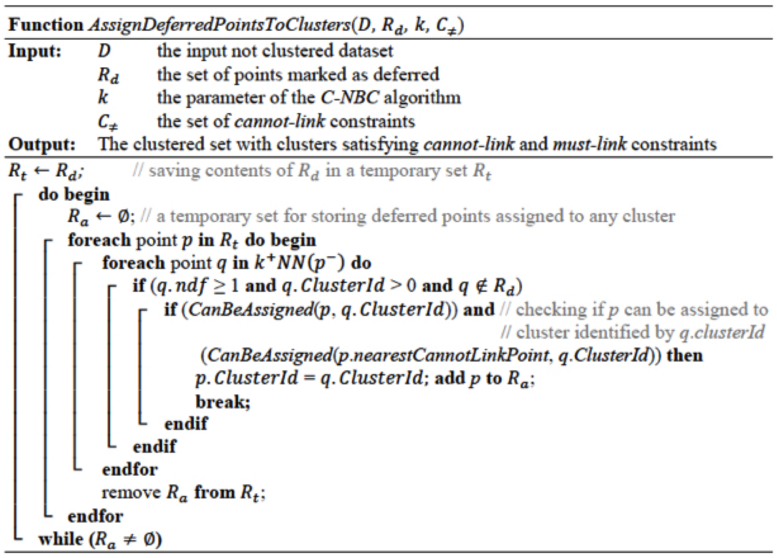
Finally, other methods are based on *Density Based Spatial Clustering of Applications with Noise* (DBSCAND) proposed by Ester et al. (1996). The algorithm is focused on the concept of *neighborhood* that is defined in this case as *a region of given radius and containing a minimum number of data points* [257, p. 218]. The shape of a neighborhood is defined by the choice of a distance function for two points $p$ and $q$. Different concepts and aspects are introduced inside the model: some are related to the points and their position, other



```
Algorithm C-NBC(D, k, C_=, C_≠)
Input:   D        the input not clustered dataset
         k        the parameter of the C-NBC
         C_=, C_≠ the sets of pairs of points connected by must-link or cannot-link constraints
Output:  The clustered set with clusters satisfying cannot-link and must-link constraints.

  $R_d \leftarrow \emptyset$;   $ClusterId = 0$;  // initialization of a set of deferred points $R_d$ and $ClusterId$
  label all points in D as UNCLASSIFIED;  // $p.ClusterId$ = UNCLASSIFIED
  CalcNDF(D, k);  // calculate values of NDF factor for every point in D
  for each point q involved in any constraint from $C_=$ or $C_{\neq}$ do
      label q and points in $k^+NN(q^-)$ as DEFERRED   // label points as DEFERRED
      add q to $R_d$;  // add DEFERRED points to $R_d$
  endfor
  $ClusterId = 0$;
  for each unclassified point p in D such that $p.ndf \geq 1$ do
      $p.ClusterId = ClusterId$;
      clear DPSet;
      for each point $q \in k^+NN(p^-) \setminus R_d$ do
          $q.ClusterId = ClusterId$;  // set point's cluster identifier
          if ($q.ndf \geq 1$) then add q to DPset; endif
          add all points r from $C_=(q)$ such that $r.ndf \geq 1$ to DPset;
      endfor
      while ($DPSet \neq \emptyset$) do          // expanding a currently created cluster
          s = first point in DPset;
          for each unclassified point t in $k^+NN(s^-) \setminus R_d$ do
              $t.ClusterId = ClusterId$;  // set point's cluster identifier
              if ($t.ndf \geq 1$) then add t to DPset; endif
              add all points u from $C_=(t)$ such that $u.ndf \geq 1$ to DPset;
          endfor
          remove s from DPset;
      endwhile
      $ClusterId$++;
  endfor
  label unclassified points in D as NOISE;
  AssignDeferredPointsToClusters(D, $R_d$, k, $C_{\neq}$);
```

Figure 1.4: **C-NBC algorithm, source: Lasek (2014)**



```
Function AssignDeferredPointsToClusters(D, $R_d$, k, $C_{\neq}$)
Input:   D       the input not clustered dataset
         $R_d$   the set of points marked as deferred
         k       the parameter of the C-NBC algorithm
         $C_{\neq}$  the set of cannot-link constraints
Output:  The clustered set with clusters satisfying cannot-link and must-link constraints

$R_t \leftarrow R_d$;       // saving contents of $R_d$ in a temporary set $R_t$
  do begin
      $R_a \leftarrow \emptyset$; // a temporary set for storing deferred points assigned to any cluster
      foreach point p in $R_t$ do begin
          foreach point q in $k^+NN(p^-)$ do
              if ($q.ndf \geq 1$ and $q.ClusterId > 0$ and $q \notin R_d$)
                  if (CanBeAssigned(p, q.ClusterId)) and // checking if p can be assigned to
                                                         // cluster identified by q.clusterId
                     (CanBeAssigned(p.nearestCannotLinkPoint, q.ClusterId)) then
                     $p.ClusterId = q.ClusterId$; add p to $R_a$;
                     break;
                  endif
              endif
          endfor
          remove $R_a$ from $R_t$;
      endfor
  while ($R_a \neq \emptyset$)
```

Figure 1.5: **AssignDeferredPointsToClusters function, source: Lasek (2014)**

to the concept of distance between points. Specifically, it is possible to identify at least a minimum number of points neighbours, called *MinPts*, in a radius, called *Eps*. Moreover, it is possible to articulate the concept of density in [119, p. 228]:

- *directly density-reachable*: establishes that a point $p$ is directly density-reachable to a point $q$ with respect to *Eps*, *MinPts* if

  $$p \in N_{Eps} q$$

  $$\mid N_{Eps}(q) \mid \geq MinPts$$

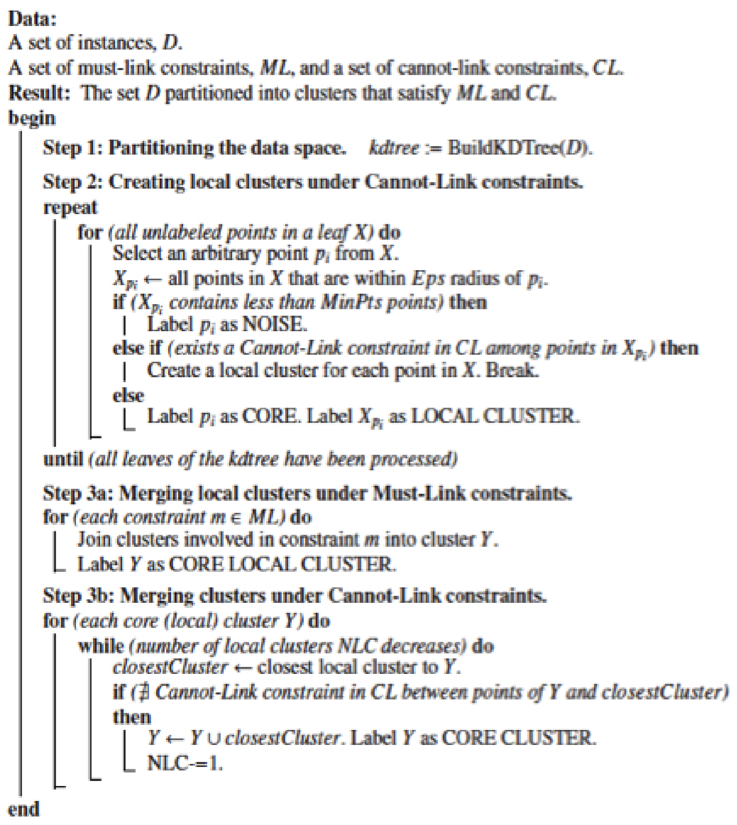
  The density-reachable is symmetric or pairs for core points.

- *density-reachable*: demonstrates that a point $p$ is density-reachable from a point $q$ with respect to Eps and *MinPts*, if it is possible to identify a chain of points $p_1, \ldots, p_n$, $p_n = q$, such that $p_{i+1}$ is directly density-reachable from $p_i$. It is a transitive relation, but it is not symmetric.

- *density-connected*: establishes that a point $p$ is density connected to a point $q$ with respect to Eps and *MinPts*, if it is possible to identify a point such that both $p$ and $q$ are density-reachable from o with respect to *Eps* and *MinPts*. In this case, the density is a symmetric relation.

Ruiz et al. (2007) have proposed another model based on different assumptions. It is called *Constraint-driven DBSCAN* (C-DBSCAN) [179]. The algorithm is composed by three phases, as shown in Figure 1.6 [257]. Firstly, the space is partitioned through the KD-Tree construction algorithm. In fact, the data space is shared iteratively into cubes by splitting planes perpendicular to the axes. The cubes become the nodes and they are partitioned in order to contain a minimum number of data points. Secondly, cannot link constraints are used to identify local clusters. In particular, if inside the leaf nodes of the KD-Tree it is identified a cannot-Link constraint, the leaf become a singleton local cluster. Otherwise, if there is no cannot-Link constraints, the algorithm evolves as a traditional DB-SCAN . It is firstly checked the presence of neighborhood with at least a minimum point. If the neighborhood has too few points, then $p$ is considered as a noise point and is therefore ignored. Otherwise, all data points, which are density-reachable from it, become members of the same local cluster. In the final step, the algorithm joins together the adjacent clusters hierarchically, while enforcing the remaining Cannot-Link constraints.

To conclude, the semi supervised learning approach has been described in order to show how this method combines the advantage of both the supervised and the unsupervised



```
Data:
A set of instances, $D$.
A set of must-link constraints, $ML$, and a set of cannot-link constraints, $CL$.
Result: The set $D$ partitioned into clusters that satisfy $ML$ and $CL$.
begin
    Step 1: Partitioning the data space.    $kdtree$ := BuildKDTree($D$).
    Step 2: Creating local clusters under Cannot-Link constraints.
    repeat
        for (all unlabeled points in a leaf $X$) do
            Select an arbitrary point $p_i$ from $X$.
            $X_{p_i} \leftarrow$ all points in $X$ that are within $Eps$ radius of $p_i$.
            if ($X_{p_i}$ contains less than $MinPts$ points) then
                Label $p_i$ as NOISE.
            else if (exists a Cannot-Link constraint in $CL$ among points in $X_{p_i}$) then
                Create a local cluster for each point in $X$. Break.
            else
                Label $p_i$ as CORE. Label $X_{p_i}$ as LOCAL CLUSTER.
    until (all leaves of the kdtree have been processed)
    Step 3a: Merging local clusters under Must-Link constraints.
    for (each constraint $m \in ML$) do
        Join clusters involved in constraint $m$ into cluster $Y$.
        Label $Y$ as CORE LOCAL CLUSTER.
    Step 3b: Merging clusters under Cannot-Link constraints.
    for (each core (local) cluster $Y$) do
        while (number of local clusters NLC decreases) do
            $closestCluster \leftarrow$ closest local cluster to $Y$.
            if ($\nexists$ Cannot-Link constraint in $CL$ between points of $Y$ and $closestCluster$)
            then
                $Y \leftarrow Y \cup closestCluster$. Label $Y$ as CORE CLUSTER.
                NLC-=1.
end
```

Figure 1.6: **Constraint-driven DBSCAN, source: Ruiz et al. (2007)**

learning approaches, and the labeled and unlabeled data. A review of the literature has been made to identify a framework where the proposal of this thesis will find its place.
The framework is the unsupervised learning method and, specifically, those algorithms that are able to obtain clusters associated with an outcome variable. The aim is to define an algorithm able to define groups that are similar with respect to a specific variable.
In the following two chapters, the methodologies used to define the proposal of this thesis will be presented. Specifically, the second chapter will be focused on the study of the tree-based methods, the third on the study of community detection algorithms. Finally the fourth chapter will present the algorithm and its applications.

# Chapter 2

# Tree-based methods

## 2.1 Abstract

The chapter is focused on the study of the tree-based methodologies. Origins, algorithms, strengths and weaknesses will be considered and analyzed. The aim is to analyze the different tree algorithms to identify the most useful for the development of the proposal of this thesis.
Three parts compose the chapter. The first part focuses on the definition of tree-based methods, the second on the analysis of different algorithms, their evolution and criteria and the third on the study of bagging, random forest and boosting. Finally, in the last part of the chapter, the methodologies used to define the new semi supervised algorithm is presented.

## 2.2 Tree-based methods

The tree-based methods define a wide set of methodologies finalized to partition the features space in different areas with the aim to realize classification and regression analysis [283, 101].
Generally, given a dependent variable $Y$, $n$ observations and $X_i$ covariates, the three involves splitting progressively the observations in subsets. The aim is to obtain subset more homogenous within compared to the initial set. Step by step, the group heterogeneity is reduced until it is defined the terminal node with high level of homogeneity.
When the dependent variable is qualitative, the tree is defined as *classification*; on the contrary, when the dependent variable is quantitative, it is called *regression.*
The tree can be also defined as *decision tree methods.* Murthy (1998) has defined decision trees as *a way to represent rules underlying data with hierarchical, sequential structures that recursively partition the data* [212, p. 1]. Moreover, Alpaydin (2010) has defined decision tree method as *a hierarchical model for supervised learning whereby the local region is*

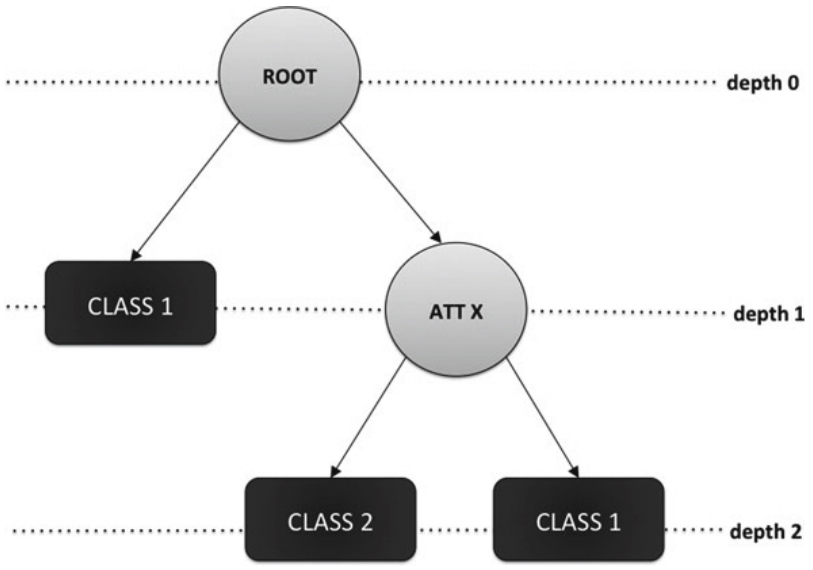


Figure 2.1: **Example of a general decision tree for classification (source: Barros et al., 2015)**

*identified in a sequence of recursive splits in a smaller number of steps* [10, p. 185]. Later, Barros et al. (2015) have stated that *decision trees are an efficient nonparametric method that can be applied either to classification or to regression tasks. They are hierarchical data structures for supervised learning whereby the input space is split into local regions in order to predict the dependent variable* [37, p. 8]. Finally, Natingga (2017) has stated that *a decision tree is the arrangement of the data in a tree structure where, at each node, data is separated to different branches according to the value of the attribute at the node* [214, p. 52].
The tree can be represented through a directed graph beginning with one node and branching to many [304]. These graphs are able to represent the partitions inside the features space.
Generally, the partitioning is realized in a top-down approach. As shown in Figure 2.1, in the first step all observations are contained in the top node. This node is later split in different nodes that contain a subset of the observations. The size of the tree is chosen through the estimation of the splitting criteria. At the end, the best tree is selected. The tree is built in order to reach different goals: firstly, to describe data through the reduction of its volume, by transforming it into a more compact form without losing the essential characteristics of the data; secondly, to provide an accurate summary of the variables; thirdly, to classify, discover and identify well-separated classes inside the data [212].
The use of trees has several advantages in many researchers' opinion. For instance, they are self-explanatory, their graphic representation is easy to understand, and is also easy

to convert them in sets of rules. Moreover, they can manage nominal and numeric input attributes, and they are also useful to analyze datasets with many errors and missing values [101, 254, 157].
At the same time, trees have also a few disadvantages, as for instance: they do not perform well if there are many complex interactions between the attributes; they are over-sensitive with respect to their training set, the irrelevant attributes and the noise [254]. Moreover, James et al. (2013) have stressed that trees do not have the same level of predictive accuracy as other regression and classification approaches. Finally, another drawback is determined by the high variance in the trees [131]. A change on the data can affect the results. The top-down process determines this instability. In fact, if some error is made in the top split, it will propagate in the whole process. A reduction of variance is recorded in Bagging that will present in the at the end of the chapter.

## 2.3 The main elements of the tree

In the building of a tree, specific elements are considered. Two are considered the main important: the definition of the splitting criterion and the stop-splitting rule.

### 2.3.1 Splitting criteria

In the building of trees, a major problem is the choice of attributes to be used in splitting the node in subsets. It is possible to define a spitting criterion as statistical indicator able to identify the best partitioning defined with respect to a specific covariates. Different criteria are theorized in literature. The splitting criteria defined in literature can be classified in *information theory-based criteria*, *distance-based criteria*, *other classification criteria* and, finally, *regression criteria* [212, 196, 37].
The rules relative to the information theory are based on Shannon's entropy [266]. Specifically, entropy is the measurement of the informational content of a message [71]. Shannon (1948) has defined the entropy of the set of probabilities $(p_o, p_1, \ldots, p_n)$ as [266, 71]:

$$K = -\sum p_i log p_j.$$

An example of a criterion based on the Shannon entropy is the method *Class-Attribute mutual information.* It was defined by Chain et al. (1995) in the attempt to maximize Shannon's entropy and to minimize the loss of information [78]. Defined $x = t$ as the partitioning of the one-dimensional feature space, and established that $x$ can take a higher or lower value than $t$, it is possible to define the amount of average mutual information equal to:

$$K(C,X) = \sum_{i=1}^{2}\sum_{j=1}^{2} p(c_i, x_j) log_2 \frac{\frac{c_i}{x_{ij}}}{p(c_i)}$$

where $C$ represents the set of pattern classes $c_1$ and $c_2$, which have a priori probabilities to equal respectively $p(c_1)$ and $p(c_2)$ [78, 264, 284, 37]. Moreover, $p(c_i, x_i)$ is the joint probability of the occurrence of $c_i$ and $x_i$, and $\frac{c_i}{x_{ij}}$ is the probability that the observation comes from the class $c_i$ given the outcome $x_i$ of event $X$ [78, 264, 284, 37].
Another measure based on Shannon's entropy is the *Information gain* [203, 196, 82, 37]. *Information gain is used to select the best feature (reducing the entropy by the largest amount) at each step of growing a decision tree* [82, p. 388]. This rule is based on the evaluation of impurity-based criteria. The level of impurity identifies the level of separability of the subset. Specifically, it is possible to define a subset as *pure*, if it contains the instances that belong to the same classes [37]. In this case, to define the entropy, it is necessary to consider two discrete random variables $X$ and $Y$ that have respectively the values $(x_0, x_1, \ldots, x_n)$ and $(y_0, y_1, \ldots, y_n)$ [121, 147, 69, 244]. Moreover, it is necessary to define $P_x(x_i)$ as the probability of the event $X = x_i$; $P_y(y_j)$ as the probability of the event $Y = y_j$; $P_{xy}(x_i, y_j)$ as the probability of the events $(X = x_i; Y = y_j)$; and finally $P_{x|y}(x_i|y_j)$ as the probability that $(X = x_i|Y = y_j)$. The entropy can be calculated for the variable $X$ as [121, 147, 69, 244, 82, 10]:

$$K(X) = -\sum P_x(x_i) log P_x(xi)$$

and for the joint variables $(X, Y)$ as

$$K(X,Y) = -\sum P_{xy}(x_i, y_j) log P_{xy}(x_i, y_j)$$

The third and last measure based on Shannon entropy is the *minimum entropy-of-error* (MEE), defined by Sa et al. (2009) [99]. The splitting criterion is based on the Shannon entropy of $E$ and it is calculated as:

$$H_y(E|L) = -[P_{-1} ln P_{-1} + P_0 ln P_0 + P_1 ln P_1]$$

where:

- $E$ is the *error random variable* associated to the errors $e_i = t(\omega(x_i)) - t(y(x_i))$. $e_i$ can assume values equal to $-2, 0, 2$. Zero corresponds to a correct decision; the value 2 is a misclassification that happens when $x_i$ class is the candidate class and the splitting rule produces the complement; and the value $-2$ the other way around;

- $L$ is a subset of $X$;

- $\omega$ is the class assignment function of $X$;
- $P_{-1} = P_y(E = -2)$, $P_1 = P_y(E = 2)$ and $P_o = P_y(E = 0) = 1 - P_{-1} - P_1$.

The second group of criteria is related to the *distance measures* [212, 37]. The concept of distance is referred to the distance between class probability distributions. It measures the separability, divergence or discrimination between classes. The first criterion included in this group is the Gini index. It is calculated as [138, 203, 212]:

$$\phi(t) = 1 - \sum_j p^2(j|t)$$

Another criterion is proposed by Breiman et al. (1984) and is called *twoing binary criterion*. Generally, the binary criteria require that the attributes have their domain split into two mutually exclusive subdomains, which allow to realize the binary split. They involve dividing the value of attribute $a_i$ in two subset that is possible to call $d_1$ and $d_2$. They are able, using binary criterion $\beta$, to obtain two exhaustive subsets and to define a division that maximizes the value $i$ selected for attribute $a_i$. It is possible to define the optimal binary split $\beta^*$ as:

$$\beta^* = max_{d_1,d_2}\beta(a_i, d_1, d_2, X, y)$$

then, the twoing binary criterion of Breiman et al. (1984) is calculated through the formula:

$$\beta^{twoing}(a_i, d_1, d_2, X, y) = 0,25 * p_{d_1} * p_{d_2} * (\sum_l^k abs(p_{yl|d_1} - p_{yl|d_2})^2)$$

where $abs(.)$ returns the absolute value [254, 196, 37].
Friedman (1977) and Rounds (1980) have proposed another measure based on the Kolmogorov-Smirnoff (KS) distance [129, 256]. This distance identifies a point $x^*$ that minimizes the risk of misclassification. Specifically, given $F_1(x)$ and $F_2(x)$ the univariate distributions of two classes, the point $x^*$, which minimizes the risk of misclassification, is the point for which:

$$D(x) =| F_1(x) - F_2(x) |$$

where $D(x^*) = max_x D(x)$. However, the marginal cumulative distributions are not known in nonparametric applications, but it is possible to estimate them through the empirical cumulative distributions $\hat{F_1}(x)$ and $\hat{F_2}(x)$ [129, 256]. In this way, the Kolmogorov-Smirnoff (KS) distance becomes:

$$D(x_j^*) = max_{x_j} \mid \hat{F_1}(x) - \hat{F_2}(x) \mid$$

where $D(x_{j*}^*) = max_j D(x_j^*)$.
Another criterion is the $\chi^2$ statistic. It measures the association between attributes and

classes comparing the observed values with those that one would expect. The aim is to find the predictive attribute with the highest degree of association to the class attribute. For this reason, it is necessary to maximized the value of $\chi^2$. Different researchers have used the $\chi^2$ as splitting criterion as for instance Minger (1989) and White et al. (1994). $\chi^2$ is calculated through the formula [203, 303, 37]:

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})^2}{E_{ij}}$$

where $O_{ij}$ is the observed number of cases with value $a_j$ and $E_{ij}$ is the expected number of cases which should have in case of association between variables. It is noticed that $\chi^2$ statistic is useful only if the number of frequencies is high. Otherwise, it becomes a poor approximation [37].

Another measure is the *mean posterior improvement* (MPI) theorized by Taylor and Silverman (1993). It gives a measure of the improvement obtained through the split. It assumes the maximum value when the individuals belonging to the same class are placed in the same partition. It is calculated through the formula [37]:

$$\beta^{MPI}(a_i, d_1, d_2, X, y) = p_{d_1} * p_{d_2} - (\sum_{l=1}^{k} (p_{.,yl} p_{d_1 \cap yl} p_{d_2 \cap yl}))$$

The last criterion was proposed by Mola and Siciliano (1997) and it refers to the index $\tau$. $\tau$ was firstly proposed by Goodman and Kruskal (1954) as a splitting rule [142, 207]. Mola and Siciliano (1997) have modified it to make it more useful in the definition of univariate trees. The $\tau$ index can be used to evaluate both each attribute individually and each possible binary split provided by grouping the values of a given attribute in $d_1$ and $d_2$. Defined $p_t(i)$ as the proportion of cases in node $t$ that present category $i$ of $X$, $p_t(j|i)$ as the proportion of cases in node $t$ that belong to class $j$ of $Y$ given that they have category $i$ of $X$, and $p_t^2(j)$ as the node proportion of all cases belonging to class $j$, the $\tau$ index for attribute it is calculated as:

$$\tau_i(Y|X) = \frac{\sum_i \sum_j p_t^2(j|i) p_t(i) - \sum_j p_t^2(j)}{1 - \sum_j p_t^2(j)}$$

and for the split as:

$$\tau_i(Y|X) = \frac{\sum_j p_{tl}^2(j|i) p_{tl} \sum_j p_{rl}^2(j|r) p_{tr} \sum_j p_t^2(j)}{1 - \sum_j p_t^2(j)}$$

where $X_s$ is the splitting variable, $l$ and $r$ are the two categories and $s$ is the split [207, 37]. Inside the class *other classification criteria*, it is possible to find different measures as for instance the *permutation statistic* theorized by Li and Dubes in 1986. They have defined a splitting criterion starting from the measure of similarity between two binary vectors.

Defined $V_1$ and $V_2$ as two binary vectors of size $N$, which can assume value 0 and 1, $S$ as the space of permutation of $V_2$, $D$ as the number of $(1,-1)$ matches between the entries of $V_1$ and the permutation of $V_2$, it is possible to define $S(V_1, V_2)$ as:

$$S(V_1, V_2) = P_r(D \leq d|H_0) - P_r(D = d|H_0)U$$

where $U$ is a random variable distributed uniformly over $[0, 1]$ and it is independent of $D$. Moreover, defined $C^{'}(i, j)$ as the pattern class of training pattern $i$ for each feature $j$ and $V_t(*, j) = [V_t(1, j)V_t(2, j) \dots V_t(N, j)]$ as the binary vector of the quantized version of the ordered feature values, the splitting criterion is calculated through the formula [188]:

$$S_t(j) =\mid S[V_t(*, j), C(*, j)] - 0.5 \mid$$

*This statistic measures the similarity between the pattern class labels and the values of the threshold jth feature by assessing the unusualness of the pattern class labels, when compared to randomly assigned labels with the number of labels from each class held constant* [188, p. 231].

Another interesting splitting criterion was theorized by Chandra et al. (2010). It has been called *distinct class based splitting measure* (DCSM). It is designed to reduce the impurity of the training patterns in each partition. DCSM is defined for a given attribute $x_j$ as [72, 37]:

$$M(x_j) = \sum_{v=1}^{V}[\frac{N^V}{N^U} * D(v)exp(D(v)) * \sum_{k=1}^{C}] \left[a_{\omega_k^v} * exp(\delta^v(1 - (a_{\omega_k^v})^2))\right]$$

Two parts compose the indicator. The former is $D(v)exp(D(v))$ where $D(v)$ denotes the number of distinct classes in partition $v$. When the number of distinct classes increases, the term $D(v)$ increases [72]. As well, the latter is $a_{\omega_k^v} * exp(\delta^v(1 - (a_{\omega_k^v})^2))]$. Specifically, $a_{\omega_k^v}$ and $\delta^v$ decreases when a decrease in impurity is recorded [72].

Finally, the fourth group of splitting criteria is composed by the regression criteria. For continuous variable $y$, the most common approach is the *mean squared error* (MSE) [37]:

$$MSE(a_i, X, y) = N_x^{-1} \sum_{j=1}^{|a|} \sum_{x_l \in v_l} (y(x_l) - \bar{y}(v_l))^2$$

The aim is to minimize the within-partition variance. The MSE can be calculated weighting each partition according to the estimated probability of an instance belonging to the given partition. It is estimated with the formula:

$$\omega MSE(a_i, X, y) = \sum_{j=1}^{|a|} p_{v_{j,\cdot}} \sum_{x_l \in v_l} (y(x_l) - \bar{y}(v_l))^2$$

Another common criterion is the *sum of absolute deviations* (SAD) that refers to the median and is calculated as:

$$SAD(a_i, X, y) = \sigma_x \sum_{j=1}^{|a|} p_{v_j,}.\sigma_{v_j}$$

where $\sigma_x$ is the standard deviation of instances in $X$ and $\sigma_{v_j}$ is the standard deviation of instances in $X_{a_i=v_j}$. The aim of this criterion is to reduce the value of variance in the phase of partition.
To sum up, in the creation of univariate decision trees it is possible to use different criteria considering different kinds of measures. The general aim is to identify the best tree and the classes inside the data.

### 2.3.2 Pruning criteria

The application of the splitting criteria on data can determine as a result a big tree, which can cause problems in the analysis of the results and in their interpretation. For this reason, researchers have defined pruning criteria. The aim of pruning criteria is to improve the understandability of the tree, reducing their size but without losing in accuracy.
The size reduction can be realized before or after the end of the tree building. The pre-pruning establishes when the growth of a tree must be stopped. Different criteria can be used in order to delete the branches that do not improve the predictive accuracy of the tree [118]. For instance, one of this criteria establishes that the size of the tree is chosen taking into account the class homogeneity. Another criterion establishes that it is necessary to consider a minimum number of instances for a non-terminal node [37]. Generally, these criteria are able to define primarily the size of the tree.
On the contrary, starting by an unpruned tree, the post-pruning techniques define criteria to remove sub-trees.
The main pruning methods are: *reduced-error pruning; pessimistic error pruning; minimum error pruning; critical-value pruning; cost-complexity pruning*; and *error-based pruning.*
*Reduced-error pruning* has been proposed by Quinlan (1987). It is a simple procedure starting from the bottom to the top of the tree, each internal node is replaced with the most frequent class to evaluate whether a reduction in term of accuracy is obtained. The accuracy is evaluated in terms of the reduction of misclassification error. The procedure stops when it is not possible to obtain an increase of accuracy.
Quinlan (1987) has proposed also the second criterion: the *pessimistic error pruning.* It is based on the idea that the error ratio estimated using the training set is not reliable enough, because is optimistically biased and can not be used to evaluate whether to prune tree or

not. For this reason, Quinlan (1994) has introduced a measure called *continuity correction for binomial distribution.* It is calculated as:

$$\varepsilon^{'}(T,S) = \varepsilon(T,S) + \frac{\|leaves(T)\|}{2\|S\|}$$

where $\varepsilon(T,S)$ is the error rate of the tree $T$ and sample $S$, and $\|$*leaves(T)*$\|$ defines the number of leaves of $T$. This criterion is a top-down approach. It operates analyzing the internal node of the tree and estimating the $\varepsilon$. Moreover, if an internal node is pruned, the nodes deriving from it are removed. For this reason, this criterion provides a relatively fast pruning.
The third criterion is the *minimum error pruning.* It was proposed by Niblett and Bratko (1987) and later modified by Cestnik and Bratko (1991). It is a bottom-up approach aims to minimize the expected error rate on an independent data set. Generally, the probability that an observation located in a node $t$ belongs to the *ith* class is estimated as:

$$p_i(t) = \frac{n_i(t) + p_{ai}m}{n(t) + m}$$

where

- $p_i(t)$ is the posterior probability;
- $p_{ai}$ is the prior probability of the *ith* class;
- $m$ identifies a parameter that determines the impact of $p_{ai}$ on the estimation of $p_i(t)$;
- $n_i(t)$ is the number of training examples of class $i$ reaching a node $t$;
- $n(t)$ the total number of examples in $t$.

Cestnik and Bratko have defined the $p_i(t)$ as *m-probability estimate.* Moreover, they have established that when a new observation is classified, the expected error rate is estimated as:

$$EER(t) = min_i\{1 - p_i(t) = min_i\left\{\frac{n(t) - n_i(t) + (1 - p_{ai} * m)}{nt + m}\right\}$$

Generally, if $m$ is very high, the pruning will be more severe.
The fourth criterion is the *Critical-value pruning* theorized by Mingers (1987). It is similar to the pre-pruning technique. It is an bottom-up approach, which initially defines a threshold, a critical value *cv*. The nodes of the tree are pruned if the splitting measure overcomes the *cv*. Later, the best tree is chosen taking into account the measure of the accuracy of the tree and the significance of the tree.
The fifth criterion is the *Cost-complexity pruning.* It has been proposed by Breiman (1984)

and is based on the concept of complexity cost. Specifically, given a subtree $T < T_{MAX}$, it is possible to define the cost-complexity measure $R_\alpha(T)$ as:

$$R_\alpha(T) = R(T) + \alpha\|T\|$$

where $\|T\|$ defines the complexity for any subtree taking into account the number of terminal node in $T$, $R(T)$ is a misclassification error and $\alpha$ is a real number that identifies the complexity parameter. The $R_\alpha(T)$ is estimated as the sum of the misclassification cost of the tree and the cost of the penalty for complexity [61]. The value of $\alpha$ influences the size of the tree. Specifically, if $\alpha$ is small, the penalty to obtain a big tree is small, consequently $T$ will be large. On the contrary, an increase of $\alpha$ determines a tree with a reduced number of terminal nodes. It is chosen the tree that $minR_\alpha(T)$.
The procedure of *Cost-complexity pruning* is composed by two phases. Firstly, a sequence of trees is generated beginning by the tree $T$, and a pruning trees that present the lowest increase in apparent error rate is defined. It is evaluated as:

$$\alpha = \frac{R(T) + R(T_i)}{\|\tilde{T}\| - 1}$$

Secondly, the best tree is chosen among the sequence based on its relative size and accuracy. Breiman (1984) have suggested to use the training set to build the trees and the pruning set to evaluate the cross validation.
Finally, the last criterion is the *error-based pruning* (EBP). It has been proposed by Quinlan (1993) and it is implemented as the default pruning strategy of C4.5 (a classification tree algorithm proposed by Quinlan in 1993). It is an evolution of pessimistic pruning and is based on the expected error rate. *The true novelty is that EBP simplifies a decision tree T by grafting a branch $T_t$ onto the place of the parent of t itself, in addition to pruning nodes* [118, p. 481]. To decide if a non-terminal node has to be replaces or not by a leaf, an expected error is calculated by using an upper confidence bound. Specifically, it is assumed that the errors in the training set are binomially distributed and that they can be calculated as:

$$EBP = \varepsilon(T,S) + Z_\alpha\sqrt{\frac{\varepsilon(T,S)(1-\varepsilon(T,S)}{\|S\|}}$$

where $\varepsilon(T,S)$, as evidenced before, is the misclassification rate of the tree $T$ and training set $S$, $Z$ defines the inverse of the standard normal cumulative distribution and $\alpha$ identifies the desired significance level. The tree is pruned, and the node substituted, taking into account the lowest value of EBP.
To sum up, the size of the tree can be decided before or after the construction of the tree. The prune activity is realized following different criteria. The general aim is to define the correct size of the tree in order to create better partition of the data.

## 2.4 Classification and regression trees models

As evidenced before, trees can be used to fit a relationship between the responce variable, $y$, and the covariates $x_i$. When the responce variable is qualitative, the tree is called *classification tree*, on the contrary, when the responce variable is a quantitative, the tree is called *regression tree*.

The classification tree is defined taking into account a qualitative variable $y$ that assumes values $1, 2, \ldots, K$ and $p$ predictors, $x_1, x_2, \ldots, x_p$. The classification tree involves partitioning the space of $X$ into $k$ disjoint sets defined as $R_1, R_2, \ldots, R_k$. The partition is realized recursively using one variable at time. A regression tree is similar to a classification tree, except that the Y assumes numerical values. The three algorithms were first studied and developed inside the Institute for Social Research at the University of Michigan [283, 210, 254]. These studies were focused on the building of a binary segmentation tree to comprehend the relationship between target and input attributes [212, 37]. The first study of this kind was published by Belson (1959) [253, 199].

The first regression model was written by Morgan and Sonquist in 1963. They have theorized the *Automatic Interaction Detection* (AID) [163, 283, 274]. AID is an algorithm for growing a binary regression tree [163, 253, 199]. It involves splitting the data in each node into two children nodes [190].

Instead, the first classification tree algorithm proposed in literature has been the *Theta AID* (THAID). It was theorized by Morgan and Messenger (1973). THAID is an extension of AID for categorical outcome [253, 199]. Its aim is to maximize the sum of the number of observations in each modal category [190]. It uses the *entropy* to evaluate the purity of the nodes. As evidenced before, this concept comes from the information theory and is the measure of impurity in data [93]. It is calculated as:

$$\phi(t) = -\sum_j p(j|t) log p(j|t)$$

where $p(j|t)$ is the proportion of class $j$ observations in node $t$ [190]. Another possible measure is the *Gini index.*

The second interesting classification tree model was proposed by Kass in 1980 and it was called *Chi-square Automatic Interaction Detector* (CHAID). CHAIS, as THAID before, was an evolution of AID and THAID [163, 304, 254, 71, 253]. It uses the p-value of the Chi-square and selects the predictor on the basis of the optimal split. It defines non-binary split and it works well with larger amounts of data [71]. It involves identifying a predictor $X$ useful to split a node. Later, other nodes are split considering the characteristics of the variables [190]. The CHAID was originally designed for classification and later extended to regression [190, 254].

One of the most well-known and used tree algorithm is the *Classification And Regression Trees* (CART). It has been proposed by Breiman et al. (1984). It is a sophisticated program for fitting trees to data [61, 283, 254, 71], a powerful method to make predictions and an important tools to support decisions [157]. *CART attempts to construct a binary decision tree by selecting the most useful variable from a set of candidate predictor variables* [25, p. 136].

CART can be defined as a binary recursive partitioning procedure able to process continuous and nominal variables as targets and predictors [61, 274]. It is called:

- *binary* because each node is split in two groups, first node is called *parent node* and the split nodes *child nodes*;
- *recursive* because the binary partitioning process can be applied over and over again;
- and, finally, the term *partitioning* is used to explain how the dataset is split into sections or partitioned.

It can be also defined as an analytical tool useful to explore such relationships between the target variable and covariates [273]. Yohannes and Hoddinott (1999) have defined it as a *nonparametric technique that can select from among a large number of variables those and their interactions that are most important in determining the outcome variable to be explained* [320, p. 4]. Questier et al. (2005) have defined it as a *statistical technique that can select from a large number of explanatory variables (x) those that are most important in determining the response variable (y) to be explained. This is done by growing a tree structure, which partitions the data into mutually exclusive groups (nodes) each as pure or homogeneous as possible concerning their response variable* [243, p. 46].

CART can be applied either as a classification tree or as a regressive tree. It involves stratifying or segmenting the predictor space [61, 157]. The beginning, all observations are contained inside a root node [320]. Later, the observations are split in groups or binary nodes that are internally more homogenous than the root node [320, 274]. To obtain this division, the algorithm considers the first variable and splits all possible points in two parts, also called *child nodes* [320, 254].

In case of regression tree, it is supposed to obtain $M$ partition $R_1, R_2, \ldots, R_M$ and to model the response as a constant $c_m$ in each region as:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in Rm)$$

it is chosen the sum of square as criterion minimization, and the best $\hat{c}_m$ is the average value of $y_i$ in region $R_M$. $\hat{c}_m$ is equal to:

$$\hat{c}_m = ave(y_i|x_i \in Rm)$$

To define the best partition, it is firstly consider a splitting variable $j$ and a split point $s$ and it is identified the pair of plane: $R_1(j, s) = \{X|X_j \leq s\}$ and $R_2(j, s) = \{X|X_j > s\}$. The $j$ and $s$ are chosen in way to minimize:

$$min_{j,s}[\min_{c_1} \sum_{x_1 \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_2 \in R_1(j,s)} (y_i - c_1)^2]$$

The process of partition is repeated. Later, the size of the tree is established referring to a decrease in sum-of-squares due to the split exceeds some threshold or through the cost complexity pruning explained before [131].
On the contrary, in case of classification tree, the CART uses three different measures of impurity: the misclassification error, the Gini index and cross-entropy or deviance. Defined this measure, the best split on the variable is chosen. It is split for which reduction in impurity is the highest [320, 274]. The algorithm defines the "best" splits on each variable attempting to obtain the best reduction of impurity for each split [207, 320, 274]. Finally, the splitting process continues until every observation is located in a terminal node [207, 320, 274].
A particularly interesting aspect of the CART algorithm is how it handles the missing values. Two different approaches are used. The first is used with categorical predictor. A new category called *missing* is creates and it is analyzed if the observation with missing value can have different behave than those with nonmissing values. The second approach is based on the construction of surrogate variables. Specifically, it is created a a list of surrogate predictors and split points. The first surrogate can be defined as *the predictor and corresponding split point that best mimics the split of the training data achieved by the primary split* [131, p. 311]. The same definition can be extent to the second, the third and the other surrogate. During the training phase or during prediction, the surrogates are used only whether primary splitting predictor is missing in order to reduce the effect of missing data. To sum up, CART is useful to analyze complex data. Moreover, it is able to show results in an informative and original way. Interactions become interpretable and understandable also by non-statisticians. Additionally, it is able to handle a wide range of response variables and the missing values in both response and explanatory variables [273].

## 2.5 Multivariate trees

The above analyzed model can be defined as *univariate.* In literature, another kind of tree is theorized: the *multivariate regression tree* (MRT). Barros (2015) has stated that decision trees with multivariate splits are harder to interpret respect to the univariate split. However, they are very useful because they allow to obtain smaller trees that are able to better represent the real-world application [149, 213, 254, 37].

In fact, the adjective "multivariate" is used to define different kinds of tree-based approaches. One is the tree built considering more attributes at the same time in the partition of the observations. Brodley and Utgoff (1992) have theorized an useful algorithm called *Linear Machine Decision Trees* (LMDT) [288, 63, 213, 111, 37]. It is useful to build a multi-class, multivariate decision tree using a top-down approach [288, 63, 111]. It involves using a linear machine *to classify the initial set of training instances by partitioning the feature space into R regions, one for each of the R observed classes. If the instances in a region are from one class, the region is assigned that class label. Otherwise the algorithm is applied recursively to the region* [111, p. 891]. LMDT presents two positive aspects [63, p. 1]. Firstly, it is an efficient approach to eliminate the noisy or irrelevant variables and to find a multivariate splits. Secondly, it is able to find a good partition [288, 63].

Brodley and Utgoff (1995) have stated that *the difference between univariate and multivariate trees is that in a univariate decision tree a test is based on just one feature, whereas in a multivariate decision tree a test is based on one or more features* [64, p. 47].

On the contrary, for De'ath (2002) the MRT is *a natural extension of univariate regression trees, with the univariate response of the latter being replaced by a multivariate response* [100, p. 1106]. In this second case, the focus is moved from the covariates to the outcome variable. The MRT is *a form of multivariate regression in that the response is explained, and can be predicted, by the explanatory variables* [100, p. 1106]. Moreover, De'ath (2002) has stated that the method can be also considered as a constrained clustering, because the output of the algorithm is composed by similar clusters. The MRT is particularly useful to make predictions, descriptions and explorations. The algorithm involves as a univariate tree. The difference is based on the impurity of the node. In MRT, the impurity is calculated summing the univariate impurity measure over the multivariate response, and the splits are defined minimizing the sums of squared distances of sites from the centroids of the nodes. At the end, the measure of the multivariate tree is established by cross validation.

Finally, the adjective "multivariate" is used by Zhang and Ye (2008) to define a tree-based approach to the analysis of multivariate ordinal data [324].

In multivariate trees, the definition of criteria and the selection of the features to be used are

different with respect to the univariate tree and to the different multivariate theorized tree. For instance, Brodley and Utgoff (1992) have identified five different approaches to reduce the number of features in the splitting nodes. However, two are the basic and most important: the *Sequential Backward Elimination* (SEE) and the *Sequential Forward Selection* (SFS). The former involves considering all features in the first step and then removing the features that determine the smallest decrease of the selected criterion. The latter involves considering one features in the first step and adding the features that allow obtaining the largest increase of selected partition criterion. In both cases, the criteria selected for the partition are the Gini index or the Information Gain Ratio.
On the contrary, De'ath (2002) has stated that it is necessary to redefine the measure of the impurity of a node for multivariate tree adding the univariate impurity measure. He has computed the sum of squares of the multivariate mean. He has established that each split has to minimize the *sums of squared distances* (SSD) of sites from the centroids of the nodes to which they belong [100]. Specifically, he has defined the sum of squares multivariate tree (SS-MRT) as [100]:

$$\sum_{ij} (x_{ij} - \bar{x_j})$$

where $x_{ij}$ is the species data for site $i$ and species $j$ and $\bar{x_j}$ is the mean.
The measure can also be calculated considering the median. De'ath (2002) has defined the *Multivariate sums of absolute deviations about the median* (LAD-MRT) as:

$$\sum_{ij} |(x_{ij} - \quad \tilde{x}_j)|$$

where $\tilde{x}_j$ is the median.
To conclude, it is possible to find splitting criteria for univariate and multivariate trees. The choice among them is related to the data, their characteristic and the aim of the analysis.

## 2.6 Bagging, Random forest and Boosting

Recently new algorithms have been proposed to realize decision trees. Among them, Bagging, Random forest and Boosting are especially interesting.
Bagging, also called Bootstrap aggregation, was proposed by Breiman (1996). It is a technique that allows to reduce the variance associated with prediction and to improve the prediction process [59, 40, 104, 283, 196]. It is based on bootstrap method where samples are drawn from the available data. The prediction method is applied on the data and the results are obtained through the average of regression and simple vote for classification [40, 283, 157]. The bootstrap method allows to obtain the reduction of the variance and it is

used to improve the accuracy of classification or regression methods [57, 283, 276]. Breiman (1996) has proposed the application of boosting to classify trees results in classifiers, which are generally competitive with any other classifier [40, 283]. It is possible to demonstrate that averaging a set of observations reduces variance on trees [157]. In fact, given a set of $n$ independent observations $(Z_1, \dots, Z_n)$ whose variance is equal to $\sigma^2$, the mean variance of the observations is equal to $\sigma^2/n$ [157]. In the same way, to reduce the variance and increase the accuracy, it is possible to use many training sets from the population and to build a separate prediction model using each training set. Finally, it is possible to average the resulting predictions [157]. In bagging, it can generate different bootstrapped training data sets and to apply the function on these sample and, finally, to average the prediction. Particularly, it is possible to calculate:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

where:

- $B$ is the training set;
- $\hat{f}$ is the applied function;
- $\hat{f}^{*b}$ is the applied function calculated on the $bth$ bootstrapped training set.

The application of bagging to decision trees allows to define $B$ regression trees using $B$ bootstrapped training sets and to average the resulting predictions [157]. In this way, the variance averaging the $B$ trees is reduced [283, 157].
To sum up, it is possible to state that bagging overcomes the high variability of decision trees and it is useful to improve the accuracy and efficiency in the estimation of decision trees. [57, 157].
Random forests is an improvement of bagging that is based on the use of de-correlated trees [60, 157, 198, 93]. The random forest (RF) is a powerful technique frequently used in data science [93]. Breiman (2002) has stated that *a random forest is a classifier consisting of a collection of tree-structured classifiers $h(x, \Theta_k), k = 1, \dots$ where the $\Theta_k$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $x$* [60, p. 6]. It can be also defined as a set of random decision trees in which each tree generated on a random subset of the data builds independent decision trees [198, 155, 93, 214]. The defining element of random forest is the fact that a set of predictor variables is selected randomly and restricted in each split, producing even more diverse trees [276, 157, 93]. In this way, it is possible to provide an efficient way to generate locally suboptimal splits that can improve the global performance of an ensemble of trees

[276, 93]. The rule for variable selection of $m$ variables out of total variables $p$, is $m = \sqrt{p}$ for classification and $m = p/3$ for regression problems randomly to avoid correlation among the individual trees:

Boosting algorithm [157]:
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set.
2. For $b = 1, 2, \ldots, B$ repeat: Fit a tree $\hat{f}^{*b}$ with $d$ splits ($d + 1$ terminal nodes) to the training data $(X, r)$;
Update $\hat{f}^{*b}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{*b}(x)$$

Update the residuals

$$r_i \leftarrow r_i \hat{f}^{*b}(x_i)$$

3. Output the boosted model

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^{*b}(x)$$

Box 2.1: Boosting algorithm, source James et al. (2013

Three elements are typical of the boosting model [157]:

- $B$, that indicates the number of trees. Usually, the trees are built referring to the cross-validation, to reduce overfitting;
- $\lambda$, that is a shrinkage parameter. It assumes a small positive number generally the value is equal to 0.01 or 0.001;
- $d$, that is the number of split in each tree. It can be also defined as the interaction depth, and controls interaction the interaction order. This value is really important because it is able to control the complexity of the boosted ensemble.

The distinctive aspect of Boosting is the ability to *learn slowly.* A tree is constructed only after adaptation, using residuals rather than the outcome $Y$. In this way the tree can improve the estimated function $\hat{f}$ in areas where it does not perform well [157, 198]. Moreover, shrinkage parameter slows the process down even further, allowing more and different shaped trees to attack the residuals. At the end, we obtain a model that performs well [157, 198].

## 2.7 CART and GBM

The analysis above has allowed to identify which methodologies based on trees can be useful for the definition of the algorithm. Two specific approaches are taken into account to theorize CTSC: the *Classification And Regression Trees* (CART) and the *Gradient boosting Machine* (GBM), that can be defined as an ensemble of weak prediction models, typically decision trees.

The former has been theorized by Breiman et al. (1984) and, as evidenced in chapter two, involves segmenting the predictor space [61, 157]. The observations are firstly contained in a root node. Later, they are split into groups or binary nodes that are internally more homogenous than the root node [320, 274]. The core idea is to identify a value to maximize the goodness of a split [213]. CART can be used for both qualitative and quantitative responce variable.

GBM, on the other side, has been proposed by Friedman in 2001 (Box 2.2) and is a forward stage-wise additive model [130]. It is built using the gradient descent in function space (Box 2.2) [130]. Specifically, the GBM starts defining the function able to map $x$ to $y$, $F(x)$, and minimizing the expected value of some specified loss function, which measures the discrepancy between the model and desired prediction, $L(y, F(x))$ over the joint distribution $(y, x) values$. A parameterized function $h(x, a)$, where $x$ represents the input variables and $a = \{a_1, a_2, \dots\}$ are the parameters, is taken into account in the definition of the GBM.

Moreover, an unconstrained negative gradient is included in the algorithm. This gradient defines the direction of the steepest descendent and is equal to:

$$-g_m(x_i) = \left[\frac{\partial L(y_i - F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

where $m$ is the number of iterations. Friedman (2001) has introduced a generalization of the unconstrained gradient $-g_m$ taking into account the member of the parameterized class $h(x, a_m)$ that produces $h_m = \ h(x_i, a_m)\}_1^N$ [130]. This generalization allows moving from an unconstrained to a constrained negative gradient equal to:

$$a_m = argmin_{a,\beta} \sum_{i=1}^{N} \Big[-g_m(x_i) - \beta h(x_i, a)\Big]^2$$

This constraint is used in a steepest-descent strategy. The algorithm allows also to define the successive increments as $\{\rho_m\}_1^M$. Specifically, unconstrained and constrained gradient are able to define the *steepest-descent direction*, where $\rho_m$ is called *line search* along that direction [130]. The $\rho_m$ is equal to:

$$\rho_m = argmin_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i, a_m))$$

Friedman (2001) has stated that *one can use this approach to minimize any differentiable loss $L(y, F)$ in conjunction with forward stage-wise additive modeling* [130, p. 1193]. Moreover, he has also stated that *Gradient boosting of regression trees produces competitive, highly robust, interpretable procedures for both regression and classification, especially appropriate for mining less than clean data* [130, p. 1189].

```
1. $F_0 = argmin_\rho \sum_{i=1}^{N} L(y_i, \rho)$

2. For $m = 1$ to $M$ do:

3. $\tilde{y}_i = \left[\frac{\partial L(y_i - F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

4. $a_m = argmin_{a,\beta} \sum_{i=1}^{N} \left[\tilde{y}_i - \beta h(x_i, a)\right]^2$

5. $\rho_m = argmin_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i, a_m))$

6. $F_m(x) = F_{m-1}(x) + \rho h(x, a_m)$

7. end For.
   end Algorithm.
```

Box 2.2: Algorithm Gradient Boosting

To sum up, the tree-based method is useful to support the study of the relationship between a response variable and one or more covariates. It can be an important tool to comprehend a phenomenon and to support decision-making. In literature, regression and classification trees, univariate and multivariate trees are theorized. Criteria and splitting rules are defined to improve the results of the tree and to obtain a good partition of the data. This method will be used to define the proposal of this thesis. Specifically, it will be used to analyze the data and to outline the relationship between the outcome variables and the covariates.

# Chapter 3

# Network and community detection

This chapter is focused on the study of networks and community detection algorithms. Origins, algorithms, strengths and weaknesses will be considered and analyzed.
The chapter is divided into three main parts. The first part is focused on the study of the network to define its components, links and applications. It is divided into three sections. Specifically, the first section of the first part will be dedicated to the different definitions of network and its characteristics. The second section will analyze the network representation: it will focus on the analysis of graphs in mathematical terms, identifying the key elements of networks. The third section will analyze the measure of centrality, that has been studied as a fundamental aspect to comprehend networks.
The second part on the new development in the studies of complex networks and community detection. The first section of the second part will define the main networks analyzed in literature such as: information, social, technological and biological. The second section will be focused on complex networks, their peculiarities and their models. Thirdly, the last section will define the analysis of networks in terms of communities and it will describe different useful methods to identify the communities inside the network.
Finally, the last part of this chapter is focused on the presentation of specific three community detection algorithms used to define a new semi supervised clustering method.

## 3.1 An introduction to networks

The study of networks started in the 1700s. Specifically, the first study has been realized by Leonhard Euler, a Swiss mathematician, in 1736 [120, 103]. He tried to solve a famous and old problem related to the town of Konigsberg in Prussia (the actual Lithuania). The city had been built near the mouth of the river Pregel that divides the city into four parts. To reach all the four parts of the city, citizens could use seven different bridges, shown in

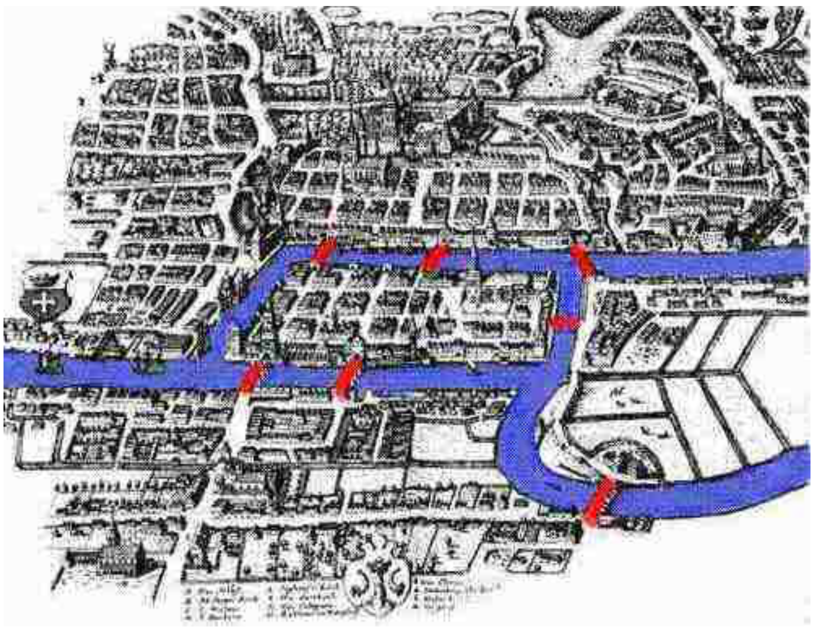


Figure 3.1: **The town of Konigsberg in Prussia, source: MacTutor History of Mathematics Archive**

Figure 3.1. Euler focused on the possibility to walk through the entire city crossing each bridge only once [9] and his study demonstrated the impossibility of this option [21].

The Euler's research has marked the birth of graph theory and the study of networks. The development of the theories and models has affected the diffusion and the application of graph theory on different contexts.

Specifically, in time researchers have provided attention on the study of *Network Theory, Theory of Network, the Network analysis* [56].

Borgatti et al. (2011) have defined network theory as *the mechanisms and processes that interact with network structures to yield certain outcomes for individuals and groups* [56, p. 1168]. The focus here is on the ability to provide a useful framework to discuss interactions among the nodes of the network.

The Theory of Network is described as the processes that determine the reason why networks have a specific structure.

Finally, the network analysis is defined by Scott (2007) as *a body of qualitative measures of network structure* [263, p. 3]. Moreover, Knobe and Yanf (2008) have defined it as *an institutionalized, transdisciplinary perspective whose basic concepts and measures are now widely familiar to researchers from such diverse fields as sociology, anthropology, economics, origination studies, business management, public health, information science, biology, complexity and chaos theory* [170, p. 2]. The development of these studies has been supported by the improvement of computing powers and by the possibility to study the properties of

a plenty of large databases of real networks [217, 277].
The use of network theory has been extended in different area such as sociology, social psychology, mathematics, political science, communication, anthropology, economics and epidemiology [217, 139, 164, 170, 125, 192, 98, 92, 259, 258]. In time, researchers have focused on the analysis of single small graphs and the properties of individual vertices or edges; as well as in the study of complex networks and community detection [219, 50].

## 3.2 The Network

Networks can be defined as an important tool to analyze and comprehend different phenomena. In literature, it is possible to find different definitions of networks.
Generically speaking, researchers have defined the network as a collection of objects in which some pairs of these objects are connected by links [114, 172]. Other studies have described it as an ensemble of nodes and edges [13, 9, 15, 3, 183, 238, 56, 172]. Finally, other researchers have defined the network as a graph composed by points joined together by connections [43, 114, 234]. The definitions offered by Newman (2003) and Jha et al. (2015) are particularly interesting. Newman (2003) defined the network as *a set of items, which we will call vertices or sometimes nodes, with connections between them, called edges* [219, p. 168]. Later, in 2010 he defined it as *a collection of points joined together in pairs by lines* [216, p. 109]. In addition, Jha et al. (2015) have described it as *a set of points (referred to as vertices or nodes) connected together by a set of lines (referred to as edges or links)* [159, p. 13].
Both definitions identify the elements that characterize networks and their nature.
Other interesting definitions of network consider different aspects, such as the relationship among the nodes and the social elements. In this specific case, the network is defined as a set of actors and the relations [170, 208]. Moreover, considering the social implication, Wasserman and Faust (1994) have defined network as *a finite set or sets of actor and the relation or relations defined on them* [301, p. 20]. Van Loon (2006) has defined network as *a device for organizing and conceptualizing non-linear complexity*[290, p. 307]. These definitions have highlighted the capacity of the network to represent the relationships existing among nodes [173, 241].
A more general definition is preferred in this work: the network is as a set of nodes and links that present specific common characteristics and are joined together by the existence of specific peculiarities.
In the study of networks, it is fundamental to comprehend the nature of the components, the

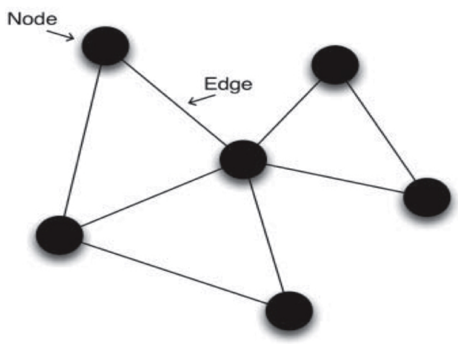


Figure 3.2: **Graph representation, source Baesens (2015)**

connections and interactions inside the network and, moreover, the pattern of the connections between the components. It is therefore necessary to define the different components of the network. These essential elements are the nodes and the links.

Van Loon (2006) has defined the nodes as the points where links are concentrated. The nature of nodes depends on the kind of network. Specifically, the nodes can be individuals or objects, biological or informatics elements, young men and women, computers or web users [192]. The different nature of nodes affects the created relationships. For instance, the nodes interact in the social networks in several ways and with a variable rate which depends on various factors [260]. The created relationships can be different: they can be defined as cooperative, hostile, predatory, competitive and aggressive [173].

The nodes are joined together by edges, as show in Figure 3.2. Edges are the most basic unit of the network. They are defined as the *line connecting two vertices* [219, p. 173]. In literature, they are often called differently: they are called *bonds* in physics, *links* in computer science, and *ties* in sociology [290]. The edges show presence of a specific relation and different level of interaction between two or more vertices [234]. Katz et al. (2004) have underlined how, in sociology, the ties can be classified in different ways such as: communication ties (who gives information or advice to whom); formal ties (who reports to whom); affective ties (who likes whom, or who trusts whom); material or work flow ties (who gives resources to whom); and finally proximity ties and cognitive ties (who knows who knows whom) [164, p. 308]. Moreover, an edge can also be defined as *directed* if it runs in only one direction; if it runs in both directions it is called *undirected.*

Another characteristic element is considered as a fundamental component by Van Loon (2006). It is the so called *mesh* and can be defined as *the overall structure, pattern and shape of the network* [290, p. 307]. This is a crucial aspect because it gives the overall structure and shapes the network: it is the distinctive element of a network and makes it unique.

To sum up, the network is characterized by three elements: the nodes, i.e. the elements that define and constitute the network; the edges, that indicate the existence and the nature of relationships between the nodes; the mesh, that identifies the structure of the network and the model of relationship.
However, these fundamental elements are not enough to define a network. It is necessary to consider two more aspects to figure out what a network is: the *walk* and the *path.*
The walk is described as a finite sequence of vertices

$$(x_o, x_1, \ldots, x_n)$$

and edges

$$(a_1, a_2, \ldots, a_n)$$

of $G$, where $G$ indicates the graph representation of the network [5, 26, 295]. Knoke and Yang (2008) define the network as *an alternating sequence of incident nodes and lines, in which each node is incident with its preceding and following lines* [170, p. 47]. Finally, it can be also defined as a sequence of lines in a graph evidencing the passage of information between the nodes [263]. A walk can be *simple* when no edge is repeated and the total number of edges cab be defined as *length of a walk* [54, 172].
The path is described as a sequence of nodes in which each consecutive pair in the sequence is connected by an edge [117, 81, 114, 172, 161]. It is a walk in which no node is visited more than once [4]. The walk of minimal length between two nodes is defined as *shortest path or geodesic* [301, 219, 172]. Moreover, it is possible to define the distance $d_{ij}$ between two vertices $i$ and $j$ as the number of links along the shortest path connecting them [43].
A graph can be *cyclic* and *acyclic.* The former, also called a *loop*, is a closed walk of at least three nodes, in which no edge is repeated [54, 26, 67, 35]. On the contrary, the latter is a graph with no cycle, also known as a forest.
An example of acyclic graph is the tree, which can therefore be defined as a connected acyclic graph [26]. Newman (2010) has defined it as *a connected, undirected network that contains no closed loops* [216, p. 127]. Finally, another element related to the path is the *diameter.* It is the length, in terms of number of edges, of the longest geodesic path between any two vertices [219].

## 3.3 Graph

It is possible to generalize the definition of network through mathematics and graph representations. This generalization is fundamental for the network to become useful in different research areas [161, 53]. Graphs can be defined as mathematical models of network structures [103, 114, 295, 53]. The different elements of graphs can be summarized with mathematical expressions. A graph $G$ can be described as a set of vertices $V$ and a collection of pairs of vertices $E$ [54, 26, 103, 65, 105]. Specifically, the graph is indicated as:

$$G = (V, E)$$

the set of vertices of $G$ with the expression:

$$V = V(G)$$

and the set of edges as:

$$E = E(G)$$

The vertex is the fundamental unit of a network and it can be called *site* in physics studies, *node* in computer science research and *actor* in sociology [219]. As evidenced before, the different vertices are joined together by edges. Given two nodes, $x$ and $y$, the edge that links the two nodes is written as $xy$ or $(x, y)$ and the nodes are called *endpoints* [65]. If vertices $x$ and $y$ are endpoints of one edge, then $x$ and $y$ are said to be *adjacent* to each other and they are written as $x \sim y$ [263, 170, 9]. Moreover, vertices adjacent to $x$ are defined as *neighbors* of $x$ and the set of all vertices adjacent to $x$ is called the *neighborhood* of $x$, denoted by $N(x)$ [295].

Graphs have two main characteristics. The first is the *order*, that expresses the number of vertices and is indicated as $v_G$. The second is the *size*, that represents the number of the edges of the graph and is indicated as $e_G$ [26, 161].

Another element that characterizes the graph is the *degree*, or *valency* (Figure 3.3) . It is indicated as $d(x)$ of the vertex $X$ and it expresses the number of edges that have $X$ as an endpoint [54, 81, 62, 43, 230, 234, 19]. Balakrishman (1997) has defined the valency as the number of edges adjacent to the vertex. When the number of edges is zero, the degree is indicated as $d(x) = 0$ and is called *isolated vertex.* When the number of edges is 1, the degree is called *pendant* and the relative edge is called *pendant edge.* In addition, if all edges have the same degree, the graph is called regular [261, 295]. Generally, the use of regular graphs is common in physics, chemistry and geo-spatial settings [172].

Further, a graph can be classified as *connected* if, for every pair of distinct nodes $i$ and $j$, it is possible to identify a path from $i$ to $j$; otherwise, it is called *unconnected* or *disconnected*, as shown in Figure 3.4 [117, 26, 35]. The bridge is the element that transforms a disconnected graph into a connected graph [9]. If the graph is disconnected, it is possible to partition the vertex set in two subsets identified as $V_1$ and $V_2$. The subsets do not have common elements.

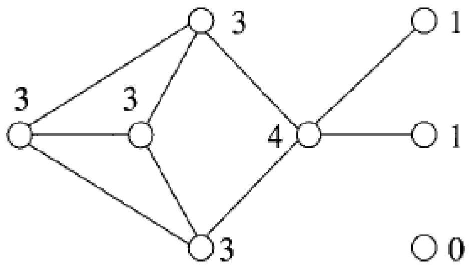


Figure 3.3: **The degrees (source: Wallis, 2007)**

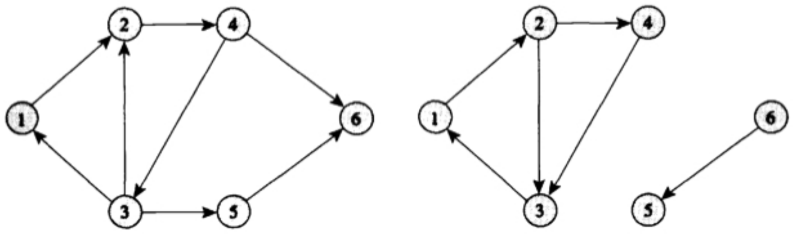


Figure 3.4: **Connected and disconnected graphs (source: Ahuja et al., 1993)**

Besides, the graph can be classified as *direct* and *indirect.* The direct graph is composed by directed edges and in this case is called digraph. In direct graph, it is important to take into account the direction of the link. Instead, if the edges are not directed in the graph, it is identified an undirected graph [54, 175].

Usually, directed edges are called *arcs* [62]. Broder et al (2000) and Wallis (2010) have defined the *arc* as an ordered pair of vertices. The first vertex of the arc is called *start, or tail or origin*, and the second is called *finish, head or terminus.* An example of direct graph is represented by the telephone calls or email messages between individuals, where each message only goes in one direction [219].
If the graph is directed, the degree of the node is obtained adding together the number of

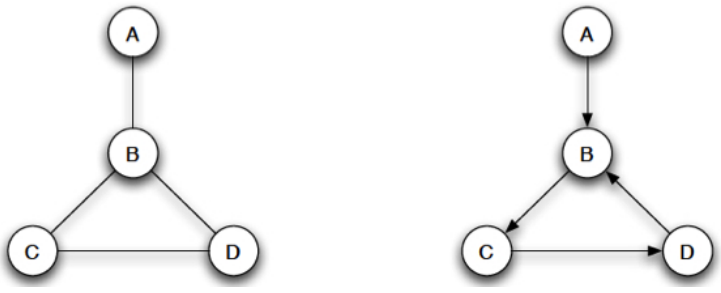


Figure 3.5: **Undirected and directed graphs (source: Easley and Kleinberg, 2010)**

outgoing links (calculated through the formula $k_i^{out} = \sum_j a_{ij}$) and the number of ingoing links ($k_i^{in} = \sum_j a_{ij}$) [234]. Consequently, the total degree is calculated through the formula $k_i = k_i^{out} + k_i^{in}$ [172]. It is possible to identify the degree distribution, $P(k)$, as the probability that a randomly chosen node has degree $k$ or, equivalently, as the fraction of nodes in the graph having degree $k$ [219, 50]. On the contrary, it is possible to define the indirect graph as a set of nodes and edges where each graph is an unordered pair of nodes $(i, j)$ [62].
Graphs can also be classified as:

- *complete*, when all nodes are connected [117, 5];
- *hypergraphs*, when edges join together more than two vertices [219];
- *isomorphic*, when two graphs have a correspondence between their vertex sets. Specifically, the isomorphic graphs have the same order and size. Given two graphs $G$ and $H$, the isomorphic graphs are indicated as $G = H$ [26].

In addition, graphs can be naturally partitioned in various ways. Specifically, the graph $H = (W, F)$ is a *subgraph* of the graph $G = (V, E)$ if $W$ and $F$ are a subset respectively of $V$ and $E$ [117, 5, 81, 261]. In other words, *the subgraph is a subset of nodes and lines within a large graph* [170, p. 47].
Graphs can also evolve over time. Their vertices or edges can appear or disappear, or values defined on those vertices and edges can change.
Finally, graphs can be represented as an adjacency matrix, i.e. a matrix $A_{n \times n}$ where the size is $nxn$ and $n$ represents the number of nodes in the network. It is indicated as $A = [a_{ij}]$, where $a_{ij} = 1$ if a link exists between node $i$ and $j$, otherwise $a_{ij} = 0$ . The matrix is symmetric when $a_{ij} = a_{ji}$ for every $i$ and $j$ [26, 49, 21, 152, 21, 11].

## 3.4 Centrality measures

In the study of networks, one of the most important topic is the centrality measures. This measures allow to comprehend how the traffic flows through a network [55, 172, 233]. They consider all the information included in the network to give better ranking results [75]. Specifically, the concept of centrality identifies the relative importance of the elements or links within the network [91]. A strong attention is necessary in the study of centrality measures in the social network analysis [233]. In time, researchers have defined different measures such as *Clustering Coefficient, Degree Centrality, Closeness, Betweenness.*
The *Clustering Coefficient* $C_i$ is a measure of the local density of a network. It quantifies the tendency of a network to be composed by groups [159, 234, 299]. Clustering, also called *transitivity*, is based on the idea that if it is possible to find in a network a vertex

$A$ connected with $B$, and $B$ connected to $C$. Consequently, the probability that the vertex $A$ will be connected with $C$ is very high. In terms of social network, this probability can be explained with the idea that a friend of your friend is likely also to be your friend [219]. Krause (2007) has described the *Clustering Coefficient* as the degree to which an individual is connected to immediate neighbors [173]. Other researchers have defined it as the ratio of the real number of the edges $E_i$ among these $k_i$ nodes to the largest number of edges, $\frac{k_i}{(k_i-1)/2}$ [36, 8, 32, 249, 67, 16]. It is calculated through the formula:

$$C_i = \frac{2E_i}{k_i(k_i-1)}$$

Assumed that it is possible to find $k_i$ nodes connected with node $i$ in the network, the $C_i$ node is defined as *neighbors of node* [297, 315].
The second centrality measure is the *Degree Centrality.* It is the number of links incident upon a node [26, 76]. It gives information about the connections of single nodes [299]. Moreover, it is a measure of the popularity of a node [223]. It is possible to identify two different measures. In fact, degree centrality can be calculated the for only the vertices or the whole graphs. In the first case, the degree centrality is calculated through the formula:

$$C_D(v) = \frac{deg(v)}{n-1}$$

where $deg(v)$ expresses the degree of the graph. In the second case, it is calculated through the formula:

$$C_D(G) = \frac{\sum_{i=1}^{\|V\|} \left(C_D(v*) - C_D(v_i)\right)}{H}$$

where $C_D(v^*)$ is the node with the highest degree centrality, $C_D(v_I)$ is the degree of centrality of the node $i$ and $H$ is the maximum value of degree. Finally, if the network is directed, two separate measures of degree centrality can be defined. Specifically, degree centrality can be defines as *indegree*, the count of the number of ties directed to the node, and as *outdegree*, the number of ties that the node directs to others [26, 263, 170].
The third centrality measure is the *Closeness*, the shortest-path length. Chen et al. (2012) have stated that the closeness is a measure of how long an information from a given node to other reachable nodes will spread into the network [75]. Wang et al. (2017) have specified that closeness is able to measure the transformation independence of a node's information in a network. They have showed how a node, if it is closer to others, can reach other nodes more easily and has a higher closeness centrality [299]. Generally, it can be stated that closeness is a centrality measure referred to the vertex within a graph: specifically, the vertices, which tend to have short geodesic distances to other vertices within the graph, have

higher closeness [75, 233]. Closeness can be calculated through the formula:

$$C_C(G) = \frac{1}{\sum_{t \in V} d_G(v,t)}$$

The denominator measures the mean geodesic distance between vertex $v$ and all other reachable vertices.

Finally, the last measure is the *Betweenness*. It can be defined considering nodes and edges. It indicates the influence and the importance of the two elements in the whole of the network [315, 299, 35]. It measures the influence of a node over the flow of information between other nodes [128, 139, 66]. Specifically, considered $n$ nodes, the *node betweenness* is the ratio between the $m$ shortest paths of nodes going through a specific node $n_i$ and the $M$ shortest paths among them [128, 76, 315]. In the same way, the *edge betweenness* is the ratio of the number of the shortest paths going through the edge to the shortest paths of all node pairs [315]. In social networks, the betweenness is defined as *the number of shortest paths between pairs of individuals that pass through a particular individual* [173, p. 17]. It is calculated through the formula [151]:

$$C_B(v) = \sum_{s \neq v \neq t \in v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where $\sigma_{st}(v)$ indicates the number of shortest paths from each pair of vertices $s$ to $t$, and $\sigma_{st}$ the number of shortest paths from $s$ to $t$ that pass through a vertex $v$ [68, 233].

## 3.5 Different kinds of networks

As evidenced before, in the last decades particular attention has been posed on the analysis of different kinds of networks. They are proposed and analyzed by researchers to comprehend their characteristic, main elements and level of complexity. Networks can be casted in four main categories: information, technological, biological and social networks [277, 219, 172].

### 3.5.1 Information Networks

The Information Networks describe the relationships among pieces of information. Two main elements are included in this group: the citation networks and the web networks.

The former is generated by the citations between academic journals or papers, as shown in Figure 3.6. Usually, researchers quote in their article previews published works. The structure of the citation network is composed by the vertices, which can be the articles, the text and the pictures held in the paper; and the links, which are the citations that join a paper to another paper [270, 8, 109, 108]. Specifically, the edges are directed from article $A$

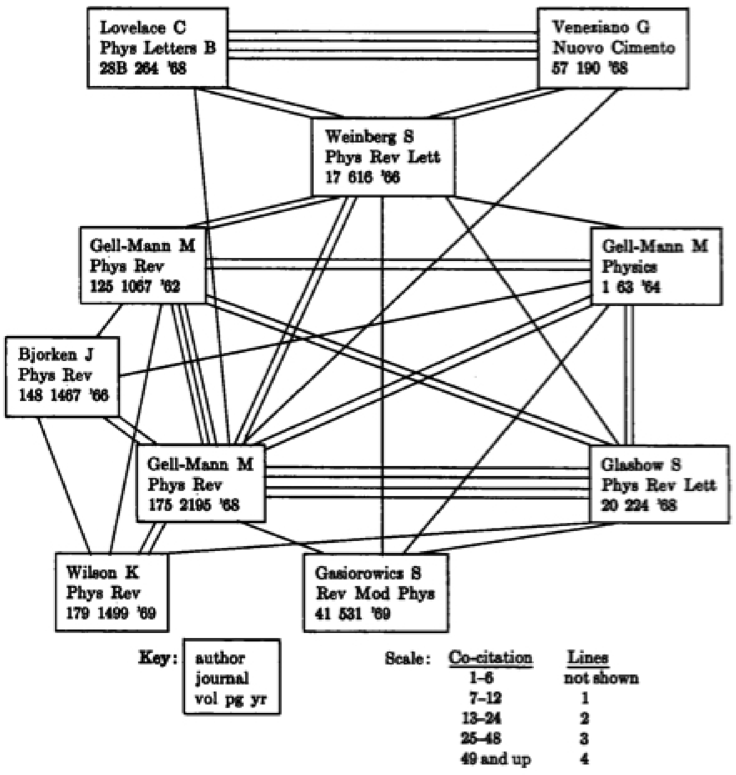

Lovelace C Phys Letters B 28B 264 '68
Veneziano G Nuovo Cimento 57 190 '68
Weinberg S Phys Rev Lett 17 616 '66
Gell-Mann M Phys Rev 125 1067 '62
Gell-Mann M Physics 1 63 '64
Bjorken J Phys Rev 148 1467 '66
Gell-Mann M Phys Rev 175 2195 '68
Glashow S Phys Rev Lett 20 224 '68
Wilson K Phys Rev 179 1499 '69
Gasiorowicz S Rev Mod Phys 41 531 '69
Key: author journal vol pg yr
Scale: Co-citation Lines
1-6 not shown
7-12 1
13-24 2
25-48 3
49 and up 4

Figure 3.6: **Citation network (source: Small, 1973)**

to article $B$ and they indicate that $A$ cites $B$ [219, 216, 311]. This network is able to show the stored information, but also the social aspect of the citation [145].
The interest in of networks of citations has started in the 1950s, when large citation databases became available [133]. One of the first research has been made by Eugene Garfield in 1955. He has proposed a citation index to improve the communication between researchers and to define a measure of the impact of papers in the research field [134, 135]. Later, other researchers have developed new studies. For instance, Small (1973) has realized the first application of citation networks. Recently, Hu et al. (2012) and Lucio-Arias and Scharnhorst (2012) have applied the graph theory to the study of citations attempting to predict the impact of papers and identify a useful mathematical approach.

The structure of Citation networks is similar to the World Wide Web (WWW). The WWW was invented in the 1980s by scientists of the CERN laboratory in Geneva as an instrument to exchange information [108, 216]. Specifically, webpages create a network in which the edges are defined by the referencing of one page by another [124, 28]. Websites as Napster or Facebook define a network where the nodes are the Internet users and the links are the exchanged content [172]. There is a big difference between the Internet and the World Wide Web: the former is composed by a physical network in which computer are

linked together by cables [28, 35]; the latter is composed by webpages, Internet users and links.
Another example of an information network is the one formed by the semantic relationships between words or concepts [8, 219]. Generally, this kind of network is analyzed in order to comprehend different aspects such as the structure of the network, which nodes are linked to many other nodes, how and how many times the structure of network changes [231, 172].

### 3.5.2 Technological Networks

The technological networks are able to represent the relationship and the distribution of resources and services. Examples of technological networks are: the communication networks, generated by telephone or internet networks; the transportation networks, generated by routes, rails and airlines routes; and the energy networks, like the circuits of gas and electricity [4, 172, 53, 161, 21].
These networks are characterized by the movement of some entities from one point to another within the underlying network. In the example above, the entities are electricity, consumer product, people and vehicles. The aim is to move these elements as soon as possible and efficiently, in order to provide a good service to the users of the network.
These networks have different kinds of ties: some are physical, as for instance cables and lines that link houses, buildings, district and countries; some are virtual connection, such as a wireless link for cellphones; finally, some are defined by the travels, as for instance the airplane routes.
The telephone network is the oldest technological network still in use [216]. On the contrary, the youngest is the Internet, which is now able to link together hundreds of millions, maybe billions of devices, as shown in Figure 3.7. Banking transactions, music, mails, videos and contents have changed completely because of these networks [219, 172].

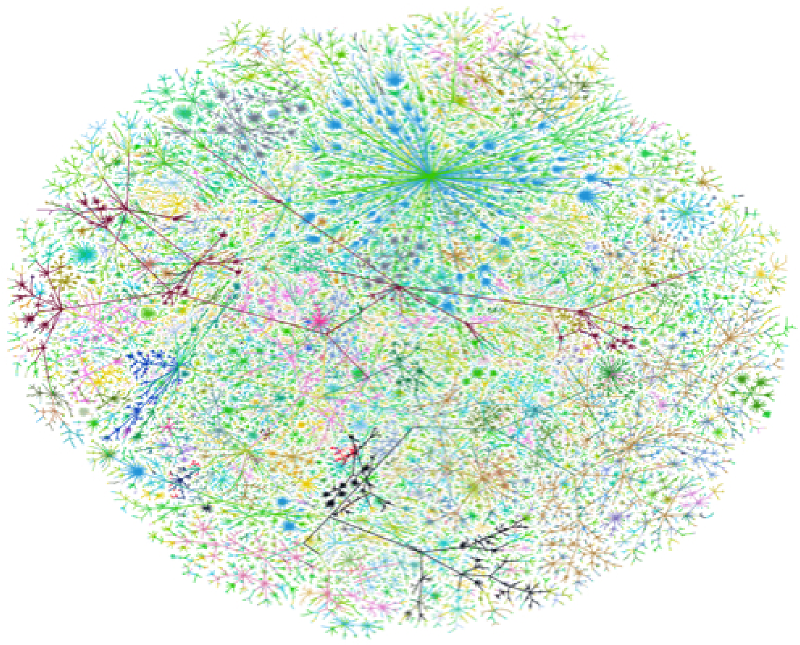


Figure 3.7: **the Internet network (source: Newman, 2006)**

### 3.5.3 Biological Networks

The third class of networks is composed by biological systems. Biologists have considered the network as a fundamental instrument to analyze the different biological phenomenon because they are able to describe the interactions between the components of a system as part of the interaction set [173]. In their studies on biological networks, researchers have focused on different areas, such as neurology, biomedicine and ecology.

Neurological networks are generated by affinity for binding among neurons [216]. Particularly interesting are the studies made by Jeong et al. (2000), that have analyzed the networks among cells.

Biomedical networks join together different kinds of networks, such as metabolic networks, protein-protein interaction networks, genetic regulatory network [8, 35, 216]. For instance, Barabasi and Oltvai (2004) have studied the interaction between genes and proteins and explained the functional organization of the cell; Ma and Zang (2003) have focused their research on metabolic networks. Other researchers have considered other biomedical topics, such as epidemiological networks, describing the spread of disease in a population.

Finally, the ecological networks are generated by the interactions among organisms. An example is the network that describes the predator-prey relationships. They are also called *food webs*. The fundamental elements of these networks are the vertices, represented by the different species, and the edges, represented by the relationship predator-prey [8, 109, 110, 216]. Particularly interesting are the studies made by Pimm (1979,1982) and

Cohen et al. (1990), who have studied the food webs to comprehend how species interact and who eats whom. Equally relevant are the studies of Dunne et al. (2002) that have analyzed how a stable community is changed after the invasion of a new species.

### 3.5.4 Social Networks

Social networks can be defined as a set of people, or groups of people, with some pattern of contacts or interactions between them [218, 219, 89, 216, 172, 42, 262]. Therefore, the Social Network Analysis (SNA) is the study of actors and connections within a social network [87]. The SNA considers different aspects and describes different phenomena. Researchers have usually analyzed the different kinds of social behaviors to investigate the nature of the relationships. Specifically, researchers have attempted to comprehend if they are cooperative, hostile, predatory, competitive or aggressive [173]. In this analysis, one of the main aspects is the definition of the nature of edges. In fact, the edges can represent different aspects: friendships, professional relationships, exchanges of goods or money, communication patterns, romantic or sexual relationships or other types of connections [216]. The intensity, frequency and directness of interactions are also an important aspect.

Generally, researchers have focused their attention on the study of social organizations composed by human beings. The main research areas have been sociology, anthropology, psychology, business and public health [172]. Moreover, in some studies the nodes of network are animals and in this case the aim is to analyze the animal behavior [173].

To sum up, it is possible to state that the scholars are interested to analyze the structure of networks, the social interactions and the different kinds of network [301].

The first steps of SNA, that took place in the 1930s, are particularly relevant. SNA has been developed with the aim to study the relationships among social entities [263, 219, 50, 170]. The first study has been realized by Jacob Moreno in 1934 [209]. He has studied the relationship between social structures, analyzing the friendship choices within the school classrooms and identifying the connections among students. Moreover, he has invented *sociogram*, a social configuration that allows to visualize the structure of networks, the channels through which the information flows from one individual to another and how individuals influence each other. It is a graph with points and lines, particularly useful to represent the nodes and their relationships.

Later in the 1950s and 1960s, different researchers cooperated inside the Tavistock Institute and the Routledge &Kegan Paul (RKP) to produce a series of relevant studies in social sciences. For instance, Clyde Mitchell (1969) and Elizabeth Bott (1957) have focused their studies on the relationships inside communities and families [301, 89]. Mitchell (1969) has defined the use of formal mathematical ideas to study community and family networks

[58, 205].
After the 1960s, the study of social networks has grown considerably. The innovations in computing have offered the possibility to investigate large-scale networks. Mark Granovetter (1973) has focused its studies on the strength of interpersonal ties; Barry Wellman (1979) has analyzed the social networks considering the community question debate in urban sociology; Berkowitz and Wellman (1988) have studied the relationships within and between organizations; Wasserman and Faust (1994) have realized a reviews of methods used in the analysis of social networks and they have developed practical example of the application of these methods; Carrington et al. (2005) have joined together the different quantitative models and methods used in social networks to offer an important support to data analysis. Recently, the study of interactions between the social networks and the Internet has been given a lot of attention. The launch and development of social media applications such as Facebook, Flickr, Twitter have caused the birth of new networks called Social Media networks [231, 21]. They are able to join and create interactions between people located in different places, cities and countries. Moreover, they facilitate the interaction and sharing of information with different people, including relatives, co-workers, family, friends, fans, and others. Each social network can be defined considering specific aspects. For instance, Facebook is more focused on social interaction, Flickr on content sharing and Twitter on widespread social interactions for users [208].
To sum up, the social networks are able to represent the relationships between individuals that interact in different ways, situations and environments.

## 3.6 Complex Networks

The last decade has witnessed the birth of a new interest around the study of networks, particularly of complex networks [30, 66, 316, 70, 325, 49, 97]. This has happened because the models proposed in mathematical graph theory were not able to represent real networks [29, 50, 182]. Researchers have felt the need to develop new models able to represent the growth of networks and to reproduce the structural properties observed in real topologies [316]. Albert and Barabasi (1999, 2002) have identified three factors that explain this new attention on complex networks. Firstly, they have underlined how the computerization of data acquisition in all fields determined the necessity and the opportunity to analyze large databases considering the topology of various real networks. Secondly, the increased computing power have allowed the study and the investigation of networks that contain millions of nodes, making it possible to analyze aspects that could have not been considered before [8, 232, 91]. Thirdly, the study of complex networks has been affected by the necessity to comprehend the behavior of systems and the interactions and relationships among the nodes within the networks [29, 8].

The three above mentioned reasons have influenced the effort of the researchers to comprehend the structure of more complex networks. The main goal is to better comprehend the complex networks, their evolutionary mechanisms and their dynamical and functional behavior [29, 50].

In the path that led to the study of complex networks, three main steps must be taken into account. First of all, the definition of the *Complex network theory* as a subfield of statistical physics for structurally disordered, dynamically heterogeneous systems with non-trivial topology. This theory was an extension of graph theory applied to systems with high structural heterogeneity and inherently dynamical properties [232].

Secondly, the definition of *Complexity science*, given by Bocaletti et al. (2014), as *the study of systems with many interdependent components, which, in turn, may interact through many different channels* [49, p. 5].

Finally, the process of definition of the concept of *complex network*, described as *[...] graphs with large size and complex* [77, p. 1317] by Chen et al. (2007), as *networks with more complex architectures than classical random graphs with their simple Poissonian distributions of connections* by Dorogovtsev et al. (2008) [107, p. 1275], and as *a graph $G$ comprising a set of $N$ nodes (or vertices) connected by a set of $M$ links (or edges), being $k_i$ the degree (number of links) of node $i$* by Arenas et al. (2008) [16, p. 95].

In mathematical terms, it is possible to define the complex network as a graph $G(V, E)$, where $V$ identifies the set of nodes, $V = n$, and $E$ the set of edges, $E = m$ [265].

The study of complex networks has therefore become a common focus of many branches of science [325, 277, 107, 192]. A relevant amount of complex real phenomena can be represented by complex networks [35]. For instance, networks such as the World Wide Web, the Internet, basic cellular networks, neural networks, co-authorship and citation networks of scientists, and many others can be defined and called *complex* [325, 277, 66, 85, 194, 70, 16, 35, 7, 297, 97].

All complex networks have a skewed distribution of connections with many hubs, strong inhomogeneity, high clustering, as well as nontrivial temporal evolution [44]. Moreover, they are *[...] open, value-laden, directed, multilevel, multicomponent, reconfigurable systems of systems, and placed within unstable and changing environments* [49, p. 6]. In addition, they are also able to change, evolve, transform through internal and external dynamic interactions affecting the subsystems and components at both local and global scale [30, 44, 261].

Finally, many complex networks are characterized by the presence of vertices with diverse and distinct functions. It can be interesting to comprehend the structure [6].

Complex networks are able to represent the composite, heterogeneous, complicated reality in people's and business life. As stated by Boccaletti et al. (2014), they are a challenge for scientists that attempt to observe, to comprehend and to predict their multi-scale and multicomponent dynamics.

### 3.6.1 The characteristics of a complex network

The study of complex networks has been focused on the analysis of models, on the definition of their specific elements and on the analysis of their peculiarities. In particular, new concepts and measures related to these kinds of networks have been developed and proposed in the past few years [8, 16]. Particular attention has been posed in the definition of three specific characteristics: *the small-world, the scale-tree and the degree distribution* [44, 115, 235, 8, 249, 194, 98].
The *small-word* concept is related to the distance and the path between nodes. Albert and Barabasi (2002) have defined the distance between two nodes as *the number of edges along the shortest path connecting them* [8, p. 38]. In addition, some researchers have evidenced that in some real large networks it is possible to reach a specific node from another following the path with the smallest number of links [235, 90, 271, 35, 216, 297, 98]. This means that it is possible to reach a specific node in a very small number of steps [90, 271, 297]. This aspect is called small-word: it describes the possibility to identify a relatively short path between any two nodes despite the large size of a network [8, 51, 91, 28, 52, 139, 90, 161]. The studies firstly of Milgram (1967) and later of Travers and Milgram (1969) on this topic are particularly interesting and famous. Milgram (1967) has established that the distance between people in the United States is on average equal to six paths, as shown in Figure 3.8 [202]. Specifically, he has realized a survey asking people that lived in Nebraska to send letters to individuals in Boston. He has given only some information about the recipients such as: name, occupation and rough location. He has supposed that hundreds of steps were necessary to deliver the letters: however, he was surprised to record in average only six steps between sender and recipient. This result has evidenced how the distance among people is not so significant. Similar results have been obtained by Dodds et al. (2003) that have considered the dispatch of emails.

The second fundamental property of complex networks is called *scale tree.* It describes the fact that the probability distribution of the number of links per node can be defined by a power-law with a degree exponent equal to $\gamma$. It can be expressed through the formula $P(k) \approx k^{\gamma}$, where the value of $\gamma$ moves from 2 to 3 [44, 108, 113, 51, 250, 249, 52, 85, 90, 306, 271, 91, 98]. In particular, complex networks are characterized to have a highly heterogeneous distribution of degrees [98]. For this reason, when one deals with a large network, it is necessary to know that the nodes have a significant probability of having a

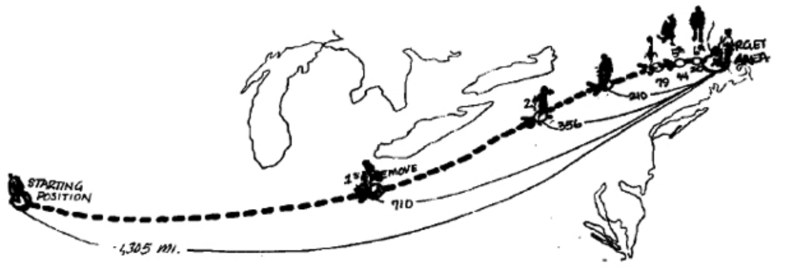


Figure 3.8: **The Milgram study, source Milgram 1967**

very large number of connections compared to the average connectivity $k$ [235]. In other words, a highly connected vertex is more likely to receive further links from newly arriving vertices [66]. Caldarelli et al. (2002) have defined this property as *rich get richer rule.* This result is totally different from what happens in the homogeneous networks where each node has approximately the same number of links [302].

The third element of complex networks is the *degree distribution.* As evidenced before, it expresses the number of edges that have X as an endpoint. The measure of degree distribution can be calculated for different kinds of networks such as bipartite graphs and direct graphs. In the first case, it is necessary to identify two-degree distributions, one for each type of vertex. In the second case, the vertex has the in-degree and the out-degree distribution at the same time. The two distributions are indicated with the function $p_{jk}$ where $j$ represents the in-degree and $k$ the out-degree distribution. The degree distribution can be represented through two histograms: the plot of the $p_k$ for any network, and the plot of the cumulative distribution function, where $p_k$ indicates the probability that the degree is greater than or equal to $k$ [219, 50]. The cumulative distribution function is given by

$$P_k = \sum_{k'=k}^{\infty} p_{k'}$$

where $P(k)$ is the distribution that gives the probability that a randomly selected node has exactly $k$ edges [8]. The study of the degree distribution is specifically related to the analysis of random graph.

In time, another property of complex networks has been introduced. It is called *fitness* [44, 28, 35]. This property refers to the probability that some nodes both have more links than other nodes and increase their degree more rapidly. This happens because some nodes have a greater fitness and they tend to win out and become very highly connected. Each node presents a different ability to compete for links. Given the fitness parameter to each node $\eta$ chosen from a distribution $p(\eta)$, the probability, which a new node connects one of its $m$ links to a node $i$, depends on the number of links $k_i$ and on the fitness of the node, and is calculated as:

$$\prod_i = \frac{\eta_i k_i}{\sum \eta_j k_j}$$

The formula evidences how new nodes link preferentially to nodes with higher $k$ and larger fitness. It is possible to state that the attractiveness and evolution of the node is determined by the fitness and the number of links [45, 44].

### 3.6.2 Complex networks models

The analysis and study of complex network have evidenced the necessity to identify new mathematical models to express and represent networks [7, 219]. In particular, in time researchers have defined models able to represent specific characteristics of the complex networks. The aim was fundamentally to explain phenomena, their characteristic and their changes in time.

Specific complex networks can be found in literature, such as *Random graphs, Generalized random graphs, Small-world networks, Static scale-free networks, Evolving scale-free networks, Weighted networks* and *Spatial networks.*

The random graph is identified as $G(n,p)$ where $n$ is the set of vertices chosen to be an edge with probability $p$ [80]. Albert and Barabãsi (2002) have defined the *Random-graph theory* as the analysis and the study of the properties of the probability space associated with graph with $N$ nodes where $N \to \infty$. Boccaletti et al. (2006) have used the term *Random network* to identify the disordered nature of the arrangement of links between different nodes. Dorogovtsev and Goltsev (2008) have defined the random graph as *a statistical ensemble, where each member is realized with some prescribed probability statistical weights* [107, p. 1277].

The study of random graph started in 1959 with the study realized by Erdös and Rényi that have attempted to identify a probabilistic method to generate random graphs with $N$ nodes and $K$ links (Figure 3.9). The model establishes that given $N$ disconnected nodes, the random graph is obtained joining together randomly selected nodes until the number of edges is equal to $K$. Albert and Barabasi (2002) have evidenced how the construction of random graphs is characterized to start with a set of $N$ isolated vertices that are joined together through random edges until a fully connected graph is obtained. The obtained random graph is only one of the many possible combinations of connections [13]. The total number of combinations is equal to $C^{n}_{[N(N-1)/2]}$ and the realization is equiprobable [7, 8, 13, 234]. The model can be also defined referring to the Binomial and the Poisson model. In the first case, the model is indicated as $G_{nk}$. It defines the graph characterized to appear with probability $p^k(1-p)^{[K-k]}$, where $K$ is the maximum possible number of edges and is calculated through the formula $K = \frac{1}{2}n(n-1)$ [109, 52, 219, 35, 234]. Instead, the Poisson model is used when $N$ is large and $k$ is fixed [50, 35]. The degree distribution follows a Poisson distribution and it is calculated as:

$$p(k) = \frac{z^k e^{-z}}{k!}$$

The model defined by Erdös and Rényi (1959) can be extended in a variety of ways to make a random graph able to represent the real networks. For example, one of this model is the *configuration model.* It was introduced by Bender and Canfield (1978) and allows to sample graphs with a given degree sequence, which is defined as a set of $n$ values of the degrees $k_i$ of vertices $i = 1, \dots, n$ from the distribution. Boccaletti at al. (2006) have defined it as *any sequence of $N$ integer numbers $D = k_1, k_2, \dots, k_N$ where $\sum_i k_i = 2K$ and $K$ is the number of links in the graph* [50, p. 192]. The model defines a process that generates every possible graphs with the given degree sequence.
The random graph is characterized by a set of nodes $n$ and a probability $p$ and they are able to represent the complexity of the real network.

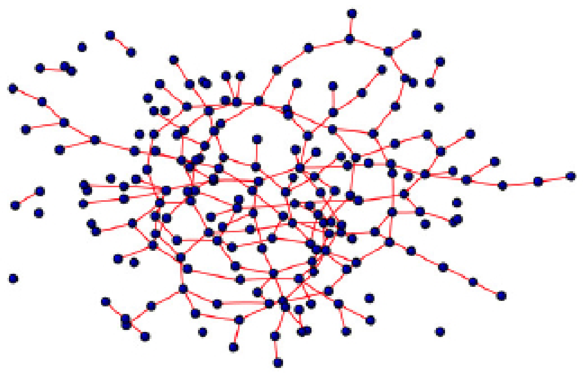


Figure 3.9: **Random graph Erdös and Rényi, source Fortunato (2010)**

The second relevant model proposed in literature is the *Small-world networks.* It was theorized by Watts and Strogatz (1998) and joins together two different models: the finite-dimensional lattices and classical random graph. The former is a variation of the Bethe lattice, that is an infinite regular graph where all vertices are topologically equivalent and boundaries are absent. The latter is the random model presented before. The combination of the two models is the base of the small-world network. It is also characterized by the combination of two properties: the small-world property and the property of high clustering coefficient. In fact, the small-world network has vertices with a specific position in space as

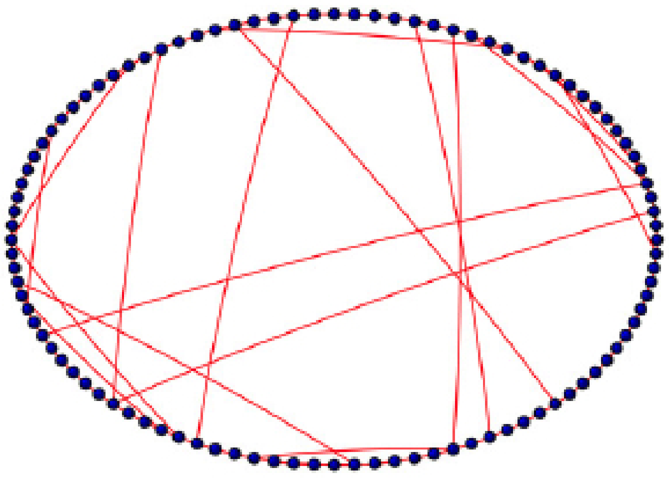


Figure 3.10: **Small-world networks Watts and Strogatz, source Fortunato (2010)**

small-world property and is based on the proximity, which plays a role in deciding which vertices are connected to others.

The starting point of the model is the presence of $N$ nodes located as a ring, where each node is connected to its nearest neighbors, as shown in Figure 3.10. Moreover, each link connected to a clockwise neighbor is the result of a rewiring procedure implemented with a probability $p$. Later, the edge is reconnected to a vertex chosen uniformly at random over the entire ring. The process is repeated by moving clockwise around the ring, considering each vertex in turn until one lap is completed. Next, the process is repeated after $k$ laps. Two quantities are typical of this model: the path length $l(p)$, which is defined as the number of edges in the shortest path between two vertices, and the cluster coefficient [34]. If $p$ is equal to zero, it obtains a regular lattice; otherwise if $p$ is equal to one, it obtains a random graph.

The small-world network presents specific properties. It is characterized by the fact that firstly local neighborhood is preserved; secondly, the diameter of the network measures average shortest distance between two vertices; and it increases logarithmically with the number of vertices $N$ [94]. Moreover, *it is possible to connect any two vertices in the network through just a few links, and the local connectivity would suggest the network to be of finite dimensionality* [12, p. 11149]. Finally, the small-world networks can be used to analyze different phenomena. For instance, as evidenced by Albert and Barabãsi (2002), this model is able to represent a social system in which the most of people are friends with their immediate neighbors and, at the same time, have some friends that live far, such as in other countries and old acquaintances.

The third model is called *static scale free network* and focuses on capturing the network dynamics [8, 13, 219]. In the reality, networks change in time and the static scale-free networks attempt to reproduce these changes. One of the most famous model was presented by Barabãsi and Albert in 1999 and is referred to indirect networks (Figure 3.11) [30, 31, 8]. The model is based on two assumptions. The former states that in the real networks the number of nodes increases in time and the networks are characterized to be an open system. The latter argues that the connection between two nodes is independent of the degree of the nodes and the new edges are placed randomly. The *static scale free network* algorithm considers the presence of a small number of nodes $m_0$ that increases at each step $t = 1, 2, 3, \dots, N$. New node $j$ and new links are added to the network. The probability that the link connects the $j$ node with another existing node $i$ is calculated with the formula:

$$\prod_{j \to 1} = \frac{k_i}{\sum k!}$$

and it is linearly proportional to the actual degree of $i$. Moreover, at each step the new node has $m$ more links. The network has $N = m_0 + t$ nodes and $K = m * t$ number of links in a specific time $t$ that corresponds to an average degree $k = 2m$ for large times. The model theorized by Barabãsi and Albert (1999) was studied by many researchers. They have attempted to build the algorithm with the aim to make the model a more realistic representation of real networks. For instance, Krapivsky and Redner have considered a directed version of the model; Dorogovtsev et al. (2000) have defined a model that considers a linear preferential attachment of the link. They have established that the probability that a link connects $j$ with another existing node $i$ with $j$ is calculated as:

$$\prod_{j \to 1} = \frac{k_i + k_0}{\sum_l k_i + k_0}$$

where $k_0$ is a constant that plays the role of the node initial attractiveness. Dorogovtsev et al. (2000) have stated that the probability is proportional to the attractiveness of the network. It is measured as $A_s = A + q_s$, where $A$ is the initial attractiveness and $A > 0$, and later the probability is equal $q_s$, which measure number of incoming links to a site $s$. Finally, Dorogovtsev and Mendes (2000) have defined a model that considers not only the possibility to add new nodes and links in a network, but also the possibility that some nodes are removed.

To summarize, the *static scale free network* is able to represent how the network changes, considering at the same time the nodes and the edges.

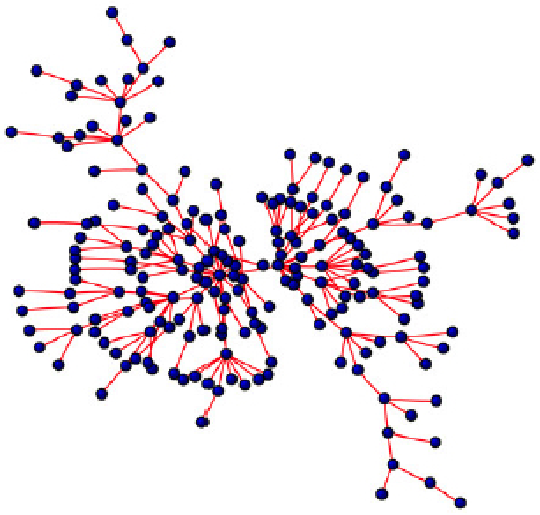


Figure 3.11: **Static scale-free networks Barabãsi and Albert, source Fortunato (2010)**

The fourth model is the *Weighted network*. It identifies and defines the existence of a different intensity of the connections between the nodes [8, 79, 80]. In this case, the different links present a numerical value that measures the strength of the connection. The presence of different levels of association has been recognized in different real networks as for instance social networks, food networks and the Internet [321, 15, 234, 50].

The weighted network can be represented through the graph $G^w = (N, L, W)$ where $N = \{n_1, n_2, \ldots, n_N\}$ indicates the set of nodes, $L = \{l_1, l_2, \ldots, l_K\}$ the set of links and $W = \{w_1, w_2, \ldots, w_K\}$ the set of weights [80, 299]. In addition, the weighted networks can be represented as a matrix $W$ where the $w_{ij}$ is the weight of the link connecting node $i$ to node $j$ [220, 50, 35].

In literature, it is possible to find different models that attempt to explain the construction of weighted networks. For instance, Boccaletti et al. (2006) have stated that the most simple model is based on random graphs with a given probability distribution $P(k)$, where the weights of the edges are distributed considering a weight distribution $Q(w)$. Yook et al. (2001) have defined the *Weighted exponential (WE) model* that is based on the unweighted scale-free networks of Barabãsi and Albert (1999). They have established that starting with $m_0$ nodes a new node $j$ is added at each step with a weight equal to:

$$w_{ij} = \frac{k_i}{\sum_{i'} k_i}$$

where $i'$ is a sum over the $m$ existing nodes to which the new node $j$ is connected. Moreover, Antal and Krapivsky (2005) have defined a model considering the edges weight. They have established that the probability the new node is connected to an already existing node is proportional to the weight of that node. The probability is equal to:

$$\prod_{j \to i} = \frac{s_i}{\sum_l k_i}$$

where $s_i$ is the node strength that is the generalization of the degree $k_i$ of a node $i$ [50].
Finally, the study of complex networks has considered other kinds of networks, as for instance those related to the space in which nodes occupy a precise position and edges are real physical connections [50]. These kinds of networks are called *Spatial networks.* It is possible to find real examples of this kind of network such as the information, communication and transportation networks [50]. Generally, these networks are geographically constrained, because the nodes are collected in spatial networks and links are limited by the physical space [50]. Many researchers have also studied spatial networks and their construction. For instance, Stoneham (1977) and Csanyi and Szendroi (2004) have studied the network with spatial implications taking into account the small-world problem. Other researchers, such as Warren et al. (2002), have analyzed the implication of scale-free structure on spatial networks. They have stated that in spatial networks the probability of connection is higher for pairs of vertex that are close together in that space [300].
The above mentioned models are able to represent the complex real networks considering specific and distinctive aspects. Each model can support the analysis of networks and the interpretation of their changes in time.

## 3.7 Community detection

In the last decade, another topic related to complex networks has been studied thoroughly: *Community Detection* [246, 83, 268, 236, 237, 225, 182, 296, 318, 252, 14, 83, 98, 18, 154, 227, 221, 185, 84, 195, 281, 312, 328]. This topic is based on the analysis of a specific characteristic of complex networks: the structure of the community [222, 291, 143, 294, 183, 307, 265, 204, 280, 41, 195]. The community structure can be defined as *a modular decomposition of the network as a number of modules that each comprises a group of nodes densely connected to each other but sparsely connected to nodes in other modules* [232, p. 3].
Following Beckett's definition (2017), the *modules* are a subset of nodes in which the interactions in the network occur within modules rather than between modules. The community structure can be also defined as a set of nodes with more internal links than external [224, 317], as shown in Figure 3.12, and the community as a set of nodes with the same properties [160, 224, 19, 299].
Many different definitions of *community* can be found in literature [323, 126]. One of them

describes it as a subgroup of nodes with a density of internal connections larger than the density of external links [139, 98, 141, 222, 127, 70, 236, 17, 291, 294, 182, 318, 152, 252, 185, 265, 14, 245, 317, 154, 227, 298, 84, 158, 126, 19, 41, 195, 312, 328, 150]. Raghavan et al. (2007) have defined it as *a group of nodes that are similar to each other and dissimilar from the rest of the network* [247, p. 1].
Communities can be found in different kinds of networks. They represent an important topic for different research areas such as social sciences, biology, computer science, engineering, economics, politics [240, 176, 98, 265, 14, 76, 204]. In each area, the community assumes a specific definition. In social networks, the community is defined as *a group of entities closer to each other in comparison to other entities of the dataset* [42, p. 116] or as individuals that interact with each other within a group more frequently than outside the group [314, 227, 42, 312].

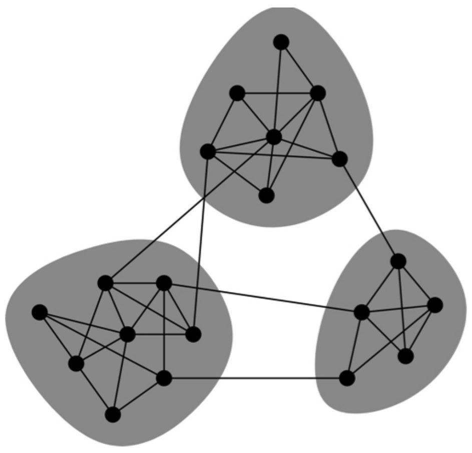


Figure 3.12: **Community in network, source: Newman, 2006**

The concept of community has been remained the same. It has been established, for instance, that communities can overlap as well as share some of the vertices, as shown in Figure 3.13 [185]. A subdivision into overlapping communities is called *cover* [126]. On the contrary, not overlapped communities are called *partitions* [126]. Instead, a cover can be defined as *crisp* when shared vertices belong to their communities with equal strength, or *fuzzy* when the strength of their membership can be different in different clusters [126].

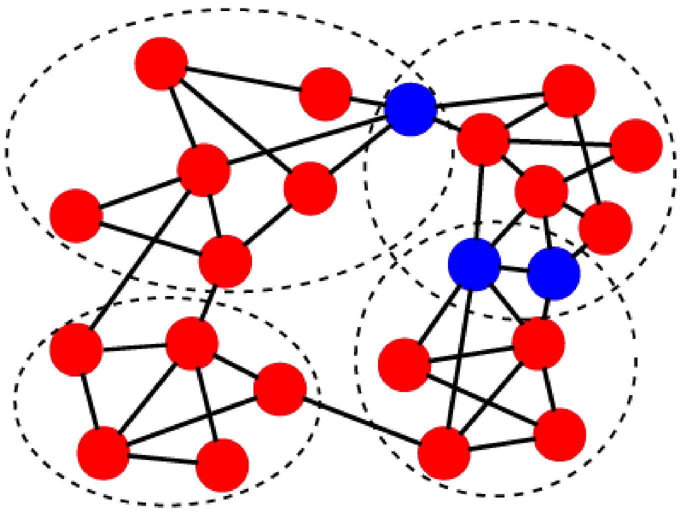


Figure 3.13: **Overlapping communities, source: Fortunato and Hric, 2016**

In addition, the network communities can be studied both at a local and a global level [261, 158]. In the first case, researchers focus their attention on the study of a unique sub graph, as for instance a subgroup composed by people that are joined by a feeling of friendship. In the second case, the focus is based on the whole graph and the aim is to comprehend the entire system of the network.

It is also possible to distinguish between strong and weak communities. Specifically, Fortunato and Hirc (2016) have defined the first as a *subgraph each of whose vertices has a higher probability to be linked to every vertex of the subgraph than to any other vertex of the graph*; the second as a *subgraph such that the average edge probability of each vertex with the other members of the group exceeds the average edge probability of the vertex with the vertices of any other group* [126, p. 7-8]. Each node has more connections within the community in strong community $V$ than with the rest of the graph, $k_i^{in}(V) > k_i^{out}(V), \forall i \in V$. On the contrary, the sum of all degrees within $V$ in weak community $V$ is larger than the sum of all degrees toward the rest of the network, $\sum_{i\in V} k_i^{in}(V) > \sum_{i\in V} k_i^{out}(V)$ [70]. It can be stated that the community in a strong sense is also a community in a weak sense, however the opposite is not true.

Community detection is an important area of the study of complex networks. Defining and identifying the community is fundamental to comprehend the structural and dynamical properties of the networks. Moreover, community detection is important to reveal both the internal organization of networks and the presence of special relationships between the nodes [224, 176, 307, 126, 231, 265, 227, 154, 245, 41]. In addition, it identifies central nodes and helps to comprehend their function and the relationships and exchanges between different communities [126, 41, 150].

Finally, the community detection aim is to divide a network into groups of nodes that are are densely connected inside but sparsely connected outside [224, 268, 98, 76, 278]. Mirsaleh and Meybodi (2016) have defined the community detection as *the process of partitioning the network into some communities in such a way that there exist many connections in the communities and few connections between them* [204, p. 535].
Applying community detection allows to divide a complex network into a number of communities characterized by a high interconnection between the nodes [224, 278, 299]. Moreover, analyzing the modular structure is important to comprehend the dynamic behavior of the system, the structure of the network and the correlation inside the network.

### 3.7.1 Community detection models

Numerous methods have been developed to realize the community detection in complex networks. Many of this methods are developed referring to tools and techniques from different disciplines such as physics, biology, applied mathematics, computer and social sciences [246, 294, 183, 176, 318, 252, 307, 231, 265, 14, 98, 265, 18, 84, 76, 281]. This has happened because the theorized algorithms are widely used in many fields [225, 265, 14, 317, 18].
Following the literature review realized by Fortunato (2010), the methods of community detection can be classified in *traditional, divisive algorithm, modularity-based methods, dynamic methods and other methods.*

### 3.7.2 Traditional methods

Traditional methods can be casted in *graph partitioning, hierarchical clustering, partitional clustering and spectral clustering* [313, 298].
In graph partitioning, the graph is divided into groups with specific properties. The number of parts is generally specified before the partition [42]. The aim of this method is to divide the vertices in $g$ groups of predefined size, such that the number of edges lying between the groups is minimal [221, 313, 216]. The number of edges running between clusters is called *cut size* [125]. One of the traditional graph partitioning method has been proposed by Kernighan and Lin (1970). It attempts to solve the following problem: *given a graph G with cost on its edge, partition the nodes of G in two subset no larger than a given maximum size, so as to minimize the total cost of the edges cut* [166, p. 291]. The method starts with an initial partition of the graph in two clusters of the predefined size. The division is realized considering some information about the graph structure, or randomly. Later, equal number of vertices is moved from one subgroup to the other to reduce the benefit function. This function measures the difference between the number of edges inside the modules and the number of edges lying between them [239, 221, 238, 216]. Finally, after some swaps with

positive and negative effect on the *benefit function*, the partition with the largest value of benefits function is selected and used as starting point of a new series of iterations. This method presents some limits as for instance the fact that is necessary to know the network and its structure before the analysis [221, 220].

Another traditional method is the *hierarchical clustering* [166, 70, 261, 318, 42]. It identifies different groups choosing a similarity measure capable of computing the similarity for each pair of vertices [221, 261, 318]. This method can be applied considering two different approaches: the agglomerative approch or the divisive one. In the first case, the similar nodes are aggregated one after the other with the aim to create a unique community. Instead, in the second case, the analysis starts considering a large cluster that is divided in smaller clusters [70, 246, 261, 42]. The clusters are obtained considering a measure of similarity $x_{ij}$ between pairs $(i, j)$ of vertices [221]. In literature, it is possible to find different measures of similarity. For instance, it is possible to use the Euclidean distance, which is calculated as:

$$x_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{ij})^2}$$

where $A_{ij}$ is an element of the adjacency matrix for vertices $i$ and $j$ [221]. The Euclidean distance assumes value zero, if the vertex pairs are equivalent in structure. Otherwise, if the pairs do not share the same neighbors, it reaches a very large value [221]. In the computation of the Euclidean distance, it is necessary to consider that *two vertices can be perfectly structurally equivalent by this measure without actually being connected to one another, the existence or not of an edge between i and j makes no difference to equation* [221, p. 325]. Another measure is the Pearson correlation between columns, or rows, of the adjacency matrix. It is calculated through the formula:

$$x_{ij} = \frac{\frac{1}{n}\sqrt{\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}}{\sigma_i \sigma_j}$$

where $\mu_i = \frac{1}{n}\sum_j A_{ij}$ and $\sigma_i^2 = \frac{1}{n}\sum_j (A_{ij} - \mu_i)^2$. This measure behaves differently: the vertices that are equivalent in terms of structure have high values in this case [221]. In both approaches, the hierarchical partitioning process can be represented by a dendrogram, as shown in Figure 3.14 [106, 83, 70, 237, 238].

The hierarchical partitioning method has the advantage that knowing many information about the network before the analysis is not necessary [221, 125]. However, it also has some disadvantages. For instance, this procedure produces many partitions and it is necessary to find the better representation of the community structure of the graph. Moreover, it is possible that vertices of a community may not be correctly classified, and vertices with just one neighbor are often classified as separated clusters in nonsense way [125]. For these

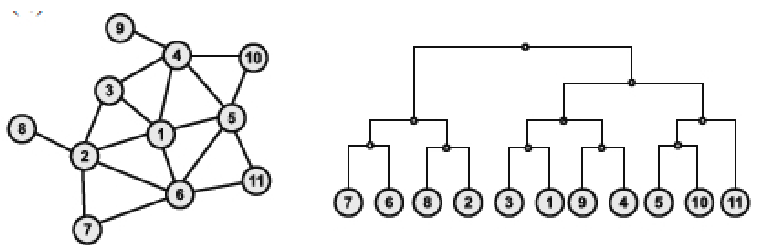


Figure 3.14: **Community in network (source: Papadopulos, 2012)**

reasons, graph partitioning is more used in computer science, whereas hierarchical clustering is more used in sociology [221, 296].

The third method proposed in literature is the *partitional clustering*. In this method, the number of clusters is pre-assigned and the point is attributed to each cluster $k$ considering the distance in the metric space. Its aim is to maximize or to minimize a given cost function based on distances between points or from points to centroids. The method starts considering the presence of centroids and it attributes each vertex to the nearest centroid. This operation is repeated until the positions of the centroids are stable. The obtained solution is not optimal, and it strongly depends on the initial choice of the centroids [125]. The most used approaches are: the *Minimum k-clustering*, where the cost function is equal to the largest distance between two points of a cluster with the aim to identify very compact clusters; the *k-clustering sum*, in which a cost function equal to the average distance between all pairs of points of a cluster is used; the *k-center*, where a centroid is defined for each cluster $i$ and the maximum of the distances of each cluster point from the centroid computed; and, the *k-median* where the maximum distance from the centroid is replaced by the average distance. Finally, the last function is the k-means clustering where the cost function is calculated as:

$$\sum_{i=1}^{k} \sum_{x_j \in s_i} ||x_j - {c_i}^2||$$

where $c_i$ is a centroid. Moreover, a variation of this method called *fuzzy k-means clustering* has been proposed, where the cost function is calculated as:

$$\sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^{m} ||x_j - {c_j}^2||$$

where $u_{ij}$ represents the membership matrix, which indicates the degree of membership of point $i$ in cluster $j$ and $c_j$ is the centre of cluster $j$. This method attempts to attribute the point at the correct cluster considering the fact that a point may belong to two or more clusters at the same time [125].

The last traditional method is the spectral clustering [261, 98, 229]. In this method, the

partition, called *cut*, into clusters is realized using the eigenvectors [221, 106, 112, 238, 318, 216, 261, 98]. To realize the clustering, the set of objects are transformed into a set of points in space, whose coordinates are elements of eigenvectors [261]. Later, the clusters are created using standard techniques such as k-means clustering. The transformation in eigenvectors allows to better represent the cluster properties [221, 125]. If the eigenvectors present high values, the groups are well separated from each other. Fundamentally, the method consists in a hierarchical clustering. It starts merging only the pairs of clusters that have at least one interconnecting edge [106, 318]. It proceeds defining different partitions, and in the end the partition with the largest modularity is chosen [106]. Finally, this model is characterized by the use of Laplacian eigenvectors with the aim to get good partitions [239, 106, 224, 238].
The first contributions about spectral clustering has been written by Donath and Hoffmann (1973) and Fiedler (1973). For what concerns the application on graphs, the model proposed by Donetti and Munoz (2004) is particularly interesting. They have defined a hierarchical clustering based approach with eigenvectors of the Laplacian matrix of the graph to find the similarity between nodes [106].

### 3.7.3 Divisive methods

The second group of methods is composed by divisive algorithms. These methods identify communities through the detection of the edges that connect vertices of different communities and removing them [229]. The aim is to disconnect the clusters from each other.
In this method, it is fundamental to choose the edges and to split the network in communities, which are constructed removing edges progressively from the original graph [139, 221, 106, 83, 70, 240, 112, 313, 238, 42, 258, 150]. The most famous method has been theorized by Girvan and Newman (2002). It is different from the previous model because it uses a new measure to define the clusters. The measure is called *betweenness*. The concept of betweenness identifies the frequency of the participation of edges to a process [139, 246, 70, 125]. Three different definitions are proposed in literature: *geodesic edge betweenness*, which identifies the number of shortest paths between all vertex pairs that run along the edge; *random-walk edge betweenness*, which is defined as the frequency of the passages across the edge of a random walker running on the graph; finally, *the current-flow edge betweenness*, that is defined as the average value of the current flow by the edge [139, 238].
The model of Girvan and Newman (2002) begins with the calculation of the betweenness and it proceeds removing the edges with the highest betweenness. Initially, the nodes are considered in a single cluster and after they are split in components. Later, the betweenness

for all edges affected by the removal is recalculated. Finally, the second phase is repeated until no edges remain. The model is based on a divisive hierarchical clustering approach [143]. The main idea of the model is that edges that run between communities have higher betweenness values than those that lie within communities. The algorithm has been applied to different real networks. Girvan and Newman (2002) have stated that their model is applicable on weighted, directed and large graphs, and is able to define communities inside the network. However, it is possible to identify some downsides, too, as for instance the absence of a guide to support the decision of how many communities a network should be split into [221].

Another divisive method has been theorised by Radicchi et al. (2004). They have defined an algorithm to discover the communities in a network considering the degree of nodes. The method operates removing the edge of lower coefficient at each step. The difference between this method and the one proposed by Newman and Girvan (2002) is that the former removes the edge with the smallest value of *edge-clustering coefficient*, whereas the latter removes the edge with the highest betweenness value. The *edge-clustering coefficient* can be defined as the fraction of triangles to which an edge can belong [246]. Given an edge that runs between two vertices $i$ and $j$, with the degrees $k_i$ and $k_j$, and assumed that is possible to identify only one edge between any pair of vertices equal to $\min(k_i - 1, k_j - 1)$, the edge-clustering coefficient can be calculated as:

$$C_{ij} = \frac{z_{ij} + 1}{\min(k_i - 1, kj - 1)}$$

where $z_{ij}$ is the measured number of triangles to which the edge belongs and +1 is a penalization to increasing weights of edges that belong to zero triangles, but which join vertices of low degree. The algorithm starts with the removal of edges with low values of $C_{ij}$; later, the coefficient is recalculated for the remaining edges. The main downside of this method is that it is totally focused on the presence of triangles in networks and if they are a few, the coefficient assumes small values, so the algorithm is not able to find the communities [246, 291, 221].

An alternative measure of centrality for edges is the *information centrality.* It is based on the concept of efficiency that measures how the information travels on a graph according to the length of the shortest paths between vertices. The information centrality is a useful measure to quantify the relevance of each edge in the network [261, 127]. Fortunato (2010) has defined it as *the average of the inverse distances between all pairs of vertices* [125, p. 100]. In addition, Fortunato et al. (2004) has defined a method based on the efficiency. In their algorithm, the centrality information is calculated estimating the distance between all pairs of vertices. The algorithm operates in four phases. It starts calculating the information centrality score for each of the edges and removing the edge with the highest score. Later, an analysis of the network components is performed and the procedure is reiterated until all edges are removed and the system breaks up into $N$ disconnected nodes. However, this method cannot be used with large-scale networks, because in this kind of networks it can be difficult to compute the shortest paths [98].

### 3.7.4 Modularity methods

The main characteristic of the third group of community detection methods is the use of a specific measure: *the modularity.* It is a measure proposed by Newman and Girvan (2004) to estimate the goodness of the modules obtained from the community detection [221, 220, 224, 33, 152, 11, 150]. The modularity is calculated through the formula:

$$Q = \sum_{l} e_{ii} - a_i^2$$

where $e_{ii}$ is the fraction of the edges that connects vertices in community $i$; $a_i = \sum_i e_{ij}$ is the fraction of edges that connects to vertices in community $i$, and $e_{ij}$ indicates the fraction of the edges connecting vertices in two different communities $i$ and $j$ [221, 222, 83]. The modularity defines the quality of a specific community division in a network [152, 105, 150]. For this reason, the best partition has a high modularity [221, 152]. Dinh and Thai (2015) have defined it as a *measure to qualify the goodness of community structures* and stated that *many efficient methods to maximize modularity have been proposed but without optimality guarantees* [105, p. 1].
Many different kinds of algorithms based on modulatory measures have been theorized. The first group is composed by *Greedy techniques* proposed by Newman (2004) [222, 294, 229]. These methods join together the vertices into communities with the aim to obtain an increase in terms of modularity. The partition is defined in different steps. Firstly, a hierarchical cluster is realized. The algorithm starts considering each singular node as a single vertex without edges. They are gradually inserted to reduce the number of the single nodes. The inclusion of vertex is realized in order to obtain the maximum increase of modularity [222]. The algorithm of Newman outperforms the previous algorithms in both modularity and efficiency [240].
Clauset et al. (2004) have proposed a different method to improve Newman's. It is based on the greedy optimization of modularity to detect communities in large networks [83]. Researchers have introduced the use of sparse matrices inside the model to rearrange the data in the form of binary trees. The algorithm starts considering each node individually

and it clusters the nodes to create a unique community. It computes the modulatory for each pair of nodes through the formula:

$$\Delta Q^{c}_{c_i,c_j} = Q(G, C - c_i - c_j + (c_i \cup c_j) - Q(\mathcal{G}, \mathcal{C})$$

where $G$ is undirected graph, $C$ is a clustering of $G$, $c_i$ and $c_j$ are two communities, $Q(\mathcal{G}, \mathcal{C})$ is the modularity of community $C$ in graph $G$. The algorithm chooses the pair of nodes that presents the maximum value of $\Delta Q$ and merges them into a new community. The algorithm stops when all nodes are joined in pair and communities [83, 294, 150].
The second groups of algorithms is called "*Simulated annealing*". It is a probabilistic procedure for the global optimization and consists in $n$ exploration of the network space to find a global optimum of the a function called *maximum* [125]. The aim is to find a stable system in which the function obtains the maximum value and it is not possible to find relevant variations of the state. This procedure was theorized firstly by Guimerã et al. (2007). They consider the network in terms of *local* and *global space*. In the first case, a single vertex is shifted randomly from one cluster to another; in the second case, the communities are merged and split. The splitting can be done in different ways. In this approach, the best results are obtained optimizing the modularity of a bipartition of the cluster [145]. Specifically, Guimerãet al. (2007) have introduced a modularity measure of bipartite networks. This kind of graphs have two partitions that are non-overlapping sets and links that have one end node from each set. An example of bipartite network is the citation network that can be shared in two sets where one represents the researchers and the other one the publications [109, 145, 325]. The same biparition can be found in the artists collaboration network, where it is possible to identify two sets: the first composed by artists, the second by teams [109, 145, 325]. Guimerã et al. (2007) have focused on the artists collaboration network and they have attempted to define groups of actors that are closely connected to each other through the concept of co-participation in many teams. Their aim was to define an approach able to identify modules in each of the two sets of nodes in the bipartite network independently.
Finally, Fortunato et al. (2010) have stated that the modularity optimization method can be used from a resolution limit problem: it is not able to classify specific cases of graphs and it determines an overestimation of network links.

### 3.7.5 Dynamic methods

Another class of methods is the *dynamic algorithms*. It is possible to include inside this group different kinds of methods based on different concepts.
The first concept is the *random walk* [261, 318]. Newman (2005) has defined the random

walk of a vertex $i$ as *the number of times that a random walk starting at s and ending at t passes through i along the way, averaged over all s and t* [223, p. 42]. The relative models are based on the idea that a strong community is characterized by the high density of internal edges: consequently, an elevate number of paths can be followed. Different researchers have used this kind of approach [318, 125]. For instance, Zhou (2003) has used random walks to define a distance between pairs of vertices; Latapy and Pons (2005) have proposed another distance measure considering the probability that the random walker moves from a vertex to another in a fixed number of steps [327]. Specifically, given a transition matrix $P_{nxn} = p_ij$ where $p_{ij} = \frac{x_{ij}}{d_i}$ and $d_i$ is the degree of $n_i$, let $P^m_{ij}$ the probability of ending at $n_j$, the distance measure $D(i, j)$ is calculated as:

$$D(i,j) = \sqrt{\sum_{n-1}^{N} \frac{(P^m_{in} - P^m_{jn})^2}{d_n}}$$

The formula can be generalized to communities as:

$$D(C_k, C_{k'}) = \sqrt{\sum_{n-1}^{N} \frac{(P^m_{C_k} - P^m_{C'_k})^2}{d_n}}$$

where $P^m_{C_k} = \frac{1}{N_k} \sum_{i\in} P^m_{ij}$ is the probability of going from $C_k$ to $n_j$, $n_i \notin C_k$ in $m$ steps.
The second concept is the *synchronization* [125]. This phenomenon describes the situation in which the units of the system are in the same or similar state at every time. Many researchers have used it to create a model for the partition of graphs, as for instance Arena et al. (2006) and Boccaletti et al. (2007) [17, 48].
The third concept is the *label propagation*, and Raghavan et al. (2007) have proposed a new method based on it [247]. They have attempted to define a method that starts giving a unique label at each vertex of the network. Later, each vertex takes the label shared by the majority of its neighbors. At the end, some labels dominate, other disappear inside the network and the communities are defined as groups of vertices having identical labels at convergence. This method has some advantages: for instance, it does not need any information about the number and the size of the clusters, and it does not need any parameter. It is interesting to underline that Raghavan et al. (2007) have stated that the community detection is similar to network partitioning. The aim of community detection is to find groups that either have an inherent or an externally specified notion of similarity among nodes within groups. Instead, the aim of network partitioning is to divide any given network into approximately equal size groups irrespective of node similarities [247].
Finally, the last concept it the map representation [255]. A cogent representation can offer important information about the network, the interactions between the elements and

the structure of complex systems. A model has been proposed by Rosvall and Bergstrom (2007). These researchers have adopted as information-theoretic approach to measure: how efficiently a map represents the underlying geography; how many details are lost in the process of simplification; and, finally, to quantify and to resolve the cartographer's tradeoff. Rosvall and Bergstrom (2007) have used a map to describe the dynamics across the links and nodes in directed and weighted networks that represent the local interactions among the subunits of a system. They focus their attention firstly on the system-wide flow of information that characterizes the behavior of the full system and, secondly, in the identification of the modules that can describe how information flows into the network. They have considered the random walk as a proxy for the information flow. They have provided a mechanism for generating a dynamic into a network capable of describing a random walk within a network. They have shown that to solve a coding problem is equivalent to find community structures in networks [255].

### 3.7.6 Other methods

It is possible to identify some models that cannot be included in the previous categories. For instance, a method has been proposed by Bagrow and Bollt (2008), an agglomerative technique called *L-shell method*. This method operates at a local level, assuming that it is possible to detect a community without requiring knowledge of the entire network. It consists in an $l-shell$ that starts from a vertex and spreads outward. Specifically, a shell is a set of vertices at a fixed geodesic distance from the origin. The method starts considering a vertex-origin and keeps adding vertices lying on successive shells. Two elements are calculated: the *emerging degree* and *total emerging degree*. The former is defined as the $l$ number of edges that connect that vertex to vertices. The latter is the sum of the emerging degrees of all vertices on the leading edge of the $l - shell$ [22]. During each iteration, the number of edges connecting vertices of the new layer to vertices inside and outside the running cluster is calculated. If the ratio of these two numbers is bigger than the predefined threshold, the vertices of the new shell are added to the cluster, otherwise the process ends [22, 125].

Another method was theorized by Eckmann and Moses (2002). It is based on a local criterion and on the use of clustering coefficient of a vertex. Specifically, this coefficient is able to distinguish tightly connected groups of vertices.

The algorithm starts with clustering. Later, the neighbors vertices and the triangle are identified inside the cluster. Then, the local curvature is defined with the aim to quantify the aggregation of triangles with congruent edges. This curvature is equal to

$$c_n = 2_{t_n}/((v_n - 1)v_n)$$

where $t_n$ is the number of triangles containing $n$ as a corner, $v_n$ is the number of links leaving $n$, and $((v_n - 1)v_n)/2$ is the maximal number of possible triangles. The graph is considered as a geometric space where the communities appear as portions of the graph with a large curvature [116].

Reichardt and Bornholdt (2006) have proposed a method that combines a quality function theorized from the same authors in (2004) and the modularity $Q$ as defined by Newman and Girvan. The quality function is calculated as [251]:

$$H(\{\sigma\}) = -\sum_{i\neq j} a_{ij}A_{ij}(\sigma_i\sigma_j) + \sum_{i\neq j} b_{ij}(1 - A_{ij})\delta(\sigma_i\sigma_j)$$

$$+\sum_{i\neq j} c_{ij}A_{ij}[1 - \delta(\sigma_i\sigma_j)] - \sum_{i\neq j} d_{ij}(1 - A_{ij})[1 - \delta(\sigma_i\sigma_j)]$$

where $A_{ij}$ is an adjacency matrix of the graph; $1, 2, \ldots, q$ are the spin state or group index of node $i$ in the graph, $a_{ij}$, $b_{ij}$, $c_{ij}$ and $d_{ij}$ are the weights of the individual contributions. If $p_{ij}$ is the probability that a link exists between node $i$ and $j$, it can be normalized to $\sum_{i\neq j} p_{ij} = 2M$, and estimated as $p_{ij} = \frac{k_i k_j}{2M}$, where $k$ is the degrees of nodes. The modularity is calculated adapting to the quality function as:

$$Q = -\frac{1}{M}H(\{\sigma\})$$

To maximize the modularity $Q$ is necessary to minimize the first function. Moreover, this algorithm uses efficient update rules to directly optimize modularity.

Another model has been proposed by Long et al. (2007). The model is based on graph approximation to learn link-pattern based community structures from a graph. Long et al. (2007) have defined a link-pattern based community *as group of nodes which have the similar link patterns* [191, p. 232]. The model generalizes the traditional graph partitioning approaches referring to different community structures. The aim of the model is to find strongly connected subgraphs from a graph.

Zarei and Samani (2009) have proposed a general spectral method to find communities. It is based on network complement and *anti-community* concepts. Specifically they have defined anti-community structure as *groups of vertices which are densely connected whereas inter-groups* [323, p. 1722]. Zarei and Samani (2009) have attempted to identify the communities in a graph looking for anticommunities in the complementary graph. They have used the eigenspace representation of matrices corresponding to a network complement to reveal the community structure of a network. They have stated that *the Laplacian eigenspace is the* *best candidate for spectral community detection especially in networks with a heterogeneous community structure* [323, p. 1722].

Another method is the *stochastic block model* [162, 197]. It is a generative model for the identification of blocks (groups). It is included inside the general class of random graph models. Generally, it involves assigning each $n$ vertices to one of $K$ blocks, groups, or communities. Later, undirected edges are located independently between vertex pairs with probability to be a function only of the group memberships of the vertices. Generally, the unnormalised maximum loglikelihood, which is able to give partition $g$ in $q$ groups of the network $G$, is defined using stochastic block model, as follows:

$$L_s(G|g) = \sum_{r,s=1}^{q} e_{r,s} log\Big(\frac{e_{r,s}}{n_r n_s}\Big)$$

where $e_{r,s}$ is the number of edges running from group $r$ to group $s$, $n_r$ and $n_s$ are the number of vertices in $r$ or $s$, and the sum runs over all pairs of groups. Karrer and Newman (2011) have modified the previouos model introducing a parameter. They have called the model *degree-corrected stochastic block model* (DCSBM) . They have defined the unnormalised maximum log-likelihood as [162]:

$$L_s(G|g) = \sum_{r,s=1}^{q} e_{r,s} log\Big(\frac{e_{r,s}}{e_r e_s}\Big)$$

where $e_r$ and $e_s$ are the sum of the degrees of the vertices in $r$ or $s$.

Finally, the last group of models is related to the benchmark [139, 177, 310, 126]. This models are based on the idea that although many algorithms have been proposed, none of them has been subjected to strict tests to evaluate their performance. For this reason, a benchmark network must be defined. It is based on a *planted - partition model* [139, 177, 310, 126]. A certain number of groups of nodes are define into network. Each node has a probability $p_{in}$ of being connected to nodes of its group and a probability $p_{out}$ of being connected to nodes of different groups [139, 177]. A groups is defined as community when $p_{in} \geq p_{out}$ inside the network. Otherwise, it is defined as a random graph when $p_{in} \leq p_{out}$. Girvan and Newman (2002) have proposed a model called the *Girvan and Newman (GN) benchmark.* The benchmark is composed by 28 nodes, each with an expected degree of 16, which are divided into four groups of 32 [139, 177, 310, 126]. Generally, GN benchmark is regularly used to test algorithms for community detection. The result of the algorithm application on different dataset can be compared with the results obtained in the application on this benchmark [139, 177, 126]. However, GN benchmark has two drawbacks. The first is that all nodes have the same expected degree, and the second is that all communities have equal size [139, 177, 126].

Lancichinetti, Fortunato and Radicchi have introduced a new class of benchmark graphs. They have generalized the GN benchmark introducing power law distributions of degree and community size [177]. It is called *Lancichinetti-Fortunato-Radicchi (LFR) benchmark*. The LFR benchmark involves different steps. Firstly, a degree $\gamma$ taken, defined as a power law distribution with exponent, is given at each node, the extremes of the distribution ($k_{min}$ and $k_{max}$) are chosen, and the average degree $k$ is defined. Secondly, *each node shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction $\mu$ with the other nodes of the network; $\mu$ is the mixing parameter* [178, p. 2]. Thirdly, the size of the communities are defined considering the power law distribution with exponent $\beta$, so that the sum of all sizes equals the number $N$ of nodes of the graph; than, a minimal and maximal community size $s_{min}$ and $s_{max}$ is established to respect $s_{min} \geq k_{min}$ and $s_{max} \geq k_{max}$. Later, all totally independent nodes are assigned to a randomly chosen community: if the community size exceeds the internal degree of the node, the node enters the community, otherwise it remains homeless, and in the next iterations it places a homeless node to a randomly chosen community. The algorithm stops when all nodes have a community. Finally, to enforce the condition on the fraction of internal neighbors expressed by the mixing parameter $\mu$, the steps are rewired to obtain the desired share $\mu$ with good approximation.

### 3.7.7 Overlapping community

Recently particular attention has been posed on the study of a special kind of complex networks: the multilayer networks. This kind of networks is called in different ways such as multilayer, multiplex and network of networks [168]. In general, it can state the multiple communities have common nodes [17, 42]. An example of multilayer community is represented by the private and personal relationships that all people develop in their life, creating a large network characterized by the interaction between communities.
Many researchers have attempted to apply the community detection to multiple communities [268, 279, 165, 97]. The aim of community detection is to divide into multiple groups that are known as overlapping nodes [268, 97]. There are specific models that can be used to evaluate this particular kind of networks. Specifically, two categories of overlapping community detection approaches have been identified. They are the node-based and the link-based algorithms [268, 165]. The former divides the nodes of the network into different communities establishing that a link usually represents the unique relation in a network. The latter operates in two phases: in the first phase, the edges of the network are clustered; in the second phase, the link communities are mapped to node communities by gathering nodes incident to all edges within each link community.

Tagarelli et al. (2017) have identified three models to analyze multilayer networks: the *flattening method*, that determines a single-layer network from the multilayer one; the *aggregation method*, that detects a community structure separately for each network layer; and finally the *direct method*, that computes a community structure directly on the input multilayer network, by optimizing some multilayer quality-assessment criterion.

## 3.8 Three community detection algorithms: *Louvain clustering*, *Walktrap* and *Label_prop*

The analysis above has allowed to identity three community detection algorithms useful to define a new semi-supervised approach. They are *Louvain*, *Walktrap* and *Label_prop*.
The *louvain clustering* has been developed by Blondel et al. (2008). It is based on *modularity* and on the method proposed by Clauset et al. (2004). Modularity is a measure theorized by Newman and Girvan (2004) to estimate the goodness of the modules obtained with community detection. It is calculated through the formula:

$$Q = \sum e_{ii} - a_i^2$$

where $e_{ii}$ is the fraction of the edges that connects vertices in community $i$; $a_i = \sum_i e_{ij}$ is the fraction of edges that connects to vertices in community $i$ and $e_{ij}$ indicates the fraction of the edges connecting vertices in two different communities $i$ and $j$. The modularity defines the quality of a particular community division in a network. For this reason, a high modularity indicates the achievement of a good partition.
The method proposed by Clauset et al. (2004) is based on the greedy optimization of modularity to detect community in large networks. Clauset et al. (2004) have introduced the use of sparse matrices inside the model to rearrange the data in the form of binary trees. The algorithm starts considering each node individually and it clusters the nodes to create a unique community. It computes the modulatory for each pair of nodes through the formula:

$$\Delta Q^c_{c_i,c_j} = Q(G, C - c_i - c_j + (c_i \cup c_j) - Q(\mathcal{G}, \mathcal{C})$$

where:

- $G = (V, E)$ identifies an undirected graph;
- $c_i$ and $c_j$ defined two different communities;
- $C$ is the clustering of $G$;

- $Q(\mathcal{G}, \mathcal{C})$ identifies the modularity

The algorithm chooses the pair of nodes that presents the maximum value of $\Delta Q$ and merge them into a new community. The algorithm stops when all nodes are joined in pairs and communities.
As evidenced before, the algorithm proposed by Blondel et al. (2008) combines modularity and the Clauset model. It is applied on weighted networks and operates in two phases. Firstly, it assigns each node of the network to different communities. In this initial partition, each node corresponds to one community. Then, the algorithm considers a node $i$ and a node $j$ and it evaluates if it is possible to record an increase in modularity joining together the two points in the same community. This step is repeated for all nodes and each node is located in the community that generates the highest increase in modularity. This process is repeated until all nodes are located and until no further improvement can be achieved. The efficiency is measured through the gain in modularity, $\Delta Q$. It is calculated by moving an isolated node $i$ into a community $C$ and through the formula:

$$\Delta Q = \{\frac{\sum_{in} + K_{i,in}}{2m} - \left(\frac{\sum_{tot} + K_i}{2m}\right)^2\} - \{\sum_{in} + K_{in} 2m - \left(\frac{\sum_{tot}}{2m}\right)^2\} - \left(\frac{K_{i,in}}{2m}\right)^2$$

where:

- $\sum_{in}$ is the sum of the weights of the links inside $C$;
- $\sum_{tot}$ is the sum of the weights of the links incident to the nodes in $C$;
- $K_i$ is the sum of the weights of the links incident to node $i$;
- $K_{i,in}$ is the sum of the weights of the links from $i$ to the nodes in $C$;
- $m$ is the sum of the weights of all the links in the network.

In the second phase, the algorithm involves building a new network, which is created using the communities found during the first phase as nodes. In this case, too, the algorithm is based on a weighted network. The weights of the links are given by the sum of the weights of the links between nodes in the corresponding two communities. When the new communities are defined, the algorithm repeats the first phase, until it is not possible to define new communities and the highest modularity is attained [47].
Equally important for the definition of the CTSC is the *cluster Walktrap*, that was proposed by Pons and Latapy (2005) and is based on the distance between vertices and on a hierarchical clustering algorithm. Pons and Latapy (2005) have introduced a new measure of distance, $r$, between the vertices, which is able to capture the community structure on

the graph. The distance is calculated considering the random walks on the graph $G$ of a given length $t$. The distance is equal to:

$$r_{ij} = \sqrt{\sum \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}}$$

where:

- $k$ indicates the number of groups in a partition;
- $t$ is the length of the random walk;
- $P^t_{ij}$ is the probability of going from the vertex $i$ and vertex $j$ through a random walk of length $t$;
- $P^t_{ik}$ is the probability for the vertex $i$ to stay on the partition $k$;
- $d(k)$ is the degree of vertices.

This distance is small when two vertices belong to the same communities, and large if they do not [237]. The formula can be generalized considering the community $C_1$ and $C_2$ as:

$$r_{C_1C_2} = \sqrt{\sum \frac{(P^t_{C_1k} - P^t_{C_2k})^2}{d(k)}}$$

The distance between two communities is calculated taking into account the random walk. Specifically, the starting vertex is chosen randomly and uniformly among the vertices of the community.
Moreover, the distance is used in the search of the communities in combination with a hierarchical cluster algorithm. The algorithm is based on an agglomerative approach and uses the Ward's method. It starts from a partition $P_1 = \{\{v\}, v \in V\}$ of the graph into $n$ communities reduced to a single vertex. Then, the distances between all adjacent vertices are calculated. At each step $k$, two communities, $C_1$ and $C_2$, in $P_k$ are chosen, where $P_k$ is a partition of the graph into communities. The algorithm merges together the communities that minimize the mean $\sigma_k$ of the square distance between each vertex and its community. This is calculated as [237]:

$$\sigma_k = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} r^2_{iC}$$

For each pair of adjacent communities $\{C_1, C_2\}$, the algorithm computes the variation $\Delta\sigma(C_1, C_2)$ of $\sigma$ and it merges together the two communities that give the lowest value of $\Delta\sigma$. After merging $(C_1, C_2)$ in a new community $C_3$, it is created a new partition $P_{k+1} = (P_{k+1} \setminus \{C_1, C_2\}) \cup \{C3\}$, which updates the distances between communities. The

algorithm stops when it is obtained $P_1 = \{V\}$ after $n - 1$ steps. In other words, the algorithm runs out when all communities are merged together. The final output is shown as a dendrogram.
The last of the above-mentioned algorithms is the *Label_prop*, based on the label propagation and on the model theorized by Raghavan et al. (2007). The algorithm focuses on the hypothesis that a node $x$ has as neighbors $x_1, x_2, \ldots, x_k$, and that each neighbor carries a label denoting the community to which it belongs to. Then, the node $x$ defines its community based on the labels of its neighbors. Raghavan et al. (2007) have hypothesized that *each node chooses to join the community to which the maximum number of its neighbors belong to, with ties broken uniformly randomly* [247, p. 4]. At the starting point, every node presents a unique label and the labels propagate through the network. After this propagation, the group of nodes densely connected share the same label. Nodes that have the same labels are grouped together as one community. Some labels dominate, other disappear inside the network. Using this process the communities are defined as groups of vertices having identical labels at convergence. It is possible to summarize the algorithm in five steps [247, p. 5]:

1. it is initialized giving labels to all nodes in the network. It is identified a node as $x$ and a community as $C$;

2. at every step, each node updates its label based on the labels of its neighbors. The process is realized iteratively;

3. the nodes are arranged in the network in a random order and set it to $X$;

4. for each $x \in X$ chosen in that specific order, let

$$C_x(t) = f(C_{x_{i1}}(t), \ldots, Cx_{im}(t), Cx_{i(m+1)}(t-1), \ldots, Cx_{ik}(t-1)).$$

   The function $f$ returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly;

5. finally, if every node has a label that the maximum number of its neighbors have, the algorithm is stopped. Differently, it is necessary to restart from the third step.

The algorithm converges when a global consensus among groups is reached and the communities are identified. This method has some advantages, as for instance the fact that it does not need any information about the number and the size of the clusters. Moreover, it does not need any parameter.
The three over mentioned community detection algorithms have been used within the new

semi supervised cluster algorithm presented in the next chapter.

To conclude, the study of networks is fundamental to comprehend the structure and the characteristics of numerous phenomena. The study of nodes, edges and of the measure of centrality supports the comprehension of networks. Moreover, the definition of models defines important tools to comprehend the reality and the complexity of specific phenomena as the World Wide Web, the social interactions and also the natural events.
Finally, the possibility to identify communities inside networks offers the opportunity to study thoroughly the existence of different groups. The definition of models able to identify these communities is fundamental not only in the study of networks, but also in the cluster analysis. In fact, the community detection models offer a new tool to identify groups that are homogeneous with respect to their components, and, at the same time, heterogeneous with respect to the other groups inside the network. New application of these models is possible, as it will be shown in the following chapter.

# Chapter 4

# A new semi supervised cluster method

## 4.1 Abstract

A new semi-supervised clustering method will be proposed in this chapter. The name of this method is *Community Detection Tree-Based Algorithm for Semi-supervised Clustering* (CTSC). As evidenced by the name, different methodologies are used to build the CTSC. Specifically, tree and community detection algorithms are combined together.
CTSC aims to identify clusters associated with an outcome variable. In other words, the goal is to define a process to obtain clusters that include internally similar observations w.r.t. the values of the outcome variable, and, at the same time, groups that differ each other for the same outcome variable.
This chapter is focused on the presentation of CTSC and is divided into two parts. The first one is focused on the analysis of the specific procedures and algorithms that are necessary to build the CTSC. The second one is focused on the presentation of the algorithm and on its application to simulated datasets and to three real datasets. Finally, the differences between the proposed algorithm and the traditional cluster methods will be presented.

## 4.2 CTSC ingredients

We consider a dataset of unlabeled data $\mathbf{X}_{n,p} = (\mathbf{x}_i, y_i)_{i\in[n]}$ composed of $p$ features of which $p-1$ are completely unlabeled whilst $y_i$ represents some supervision information. Specifically, $y_i$ represents a primary information that should be considered in the clustering process and could be either a numerical variable or a categorical one. The goal of our semisupervised clustering approach is to find a partition of the $n$ instances into $k$ disjoint groups resulting the most different one form another and, equivalently, the most homogeneous in terms of observations belonging to each individual group. In this setting, between-groups heterogeneity or within-groups homogeneity is evaluated with respect to the primary information $y_i$. The partition $\Pi_k$ into $k$ groups obtained with the proposed approach is:

$$\Pi_k = (\mathcal{X}, \mathbf{y} \mid \theta_k)$$

where $\mathcal{X}$ is a data matrix composed of the $p-1$ "unlabeled" variables and $\theta_k$ is a vector of parameters to be estimated by the clustering algorithm that allows us to find $k$ clusters that are the most internally homogeneous and the most externally heterogeneous w.r.t. $y$. In other words, $\theta_k$ derives from a set of trees built according to the algorithm described in Chapter 3 (Paragraph 3.8) that provide the input to the community detection algorithm used to define the clusters. Since the main interest is finding a partition whose adequacy is evaluated w.r.t. different values of the primary information variable $y$, $\theta_k$ corresponds to a set of split points $(s_1, \ldots, s_k)$ that allow us to partition $y$ into $k$ groups. If one concentrates on the search of the most internally homogeneous groups. the goal of the method consists in finding the optimal set of parameters $\theta_k^*$, and thus the set of splitting points $\mathbf{s}_k^* = (s_1^*, \ldots, s_k^*)$ that minimize the loss function:

$$\min_{s_k} \mathcal{L}\left[(y_i, y_{ik}) \mid s_k\right].$$

If $y$ is discrete or continuous, minimizing the loss function $\mathcal{L}(\cdot)$ corresponds to the minimization of the sum of internal deviances of $y$:

$$s_k^* = \arg\min_k \left[ \sum_i^n \sum_i^k (y_{ik} - \hat{y}_k)^2 \;\middle|\; s_k \right]$$

whilst the minimization of $\mathcal{L}(\cdot)$ corresponds to minimizing the Gini impurity:

$$s_k^* = \arg\min_k \left[ 1 - \sum_j^k p_j^2 \;\middle|\; s_k \right]$$

As evidenced before, CTSC is defined taking into account different methodologies.
Tree-based algorithms are the first methodology applied on CTSC. They are a recursive partitioning of the feature space into regions containing similar observations, as underlined in Chapter 2.
Two specific approaches are taken into account to theorize the CTSC: the *Classification And Regression Trees* (CART) and the *Gradient boosting Machine* (GBM), which is an ensemble of prediction models, typically based on decision trees. The CART has been chosen because it is able to identify partitions such that the observations with similar response values are joined together in the same group [276]. Moreover, the minimization of the loss

function is useful in the definition of similar groups inside a dataset.
Community detection methods are also used in the CTSC, specifically in the clustering phase. In fact, the CTSC transforms the clustering problem in community detection problems and the algorithm involves using community detection algorithms to realize the cluster analysis. This choice has originated by the statement of Arruda et al. (2012). They have argued that *new approaches based on networks are able to detect clusters with higher accuracy than classical graph clustering methods* [96, p. 6175]. Additionally, other researchers, as for instance de Oliveira et al. (2008) and Granell et al (2011, 2012), have stressed how the cluster definition is more accurate when community detection methods are used instead of the traditional algorithm approach.
Numerous methods have been developed to identify the community detection in complex networks. As evidenced in the Chapter 3, and following the literature review presented by Fortunato (2010), it is possible to classify the methods in: traditional and divisive algorithms, modularity-based methods, dynamic methods and other. Numerous algorithms have been proposed for each different approach. Some methods are implemented in R into the package *igraph* [47].
Three specific approaches, *Louvain*, *Walktrap* and *Label_prop*, have been chosen for the building of CTSC. It has been decided to use the *Louvain* because it defines the communities taking into account the strength of the relationship between the nodes, and because it joins together the nodes that improve this relationship. This aspect is fundamental in the definition of the CTSC, as will be shown later. Equally important for the definition of the CTSC is the *cluster Walktrap.* The *Walktrap* has been chosen because it is based on the $r$ distance that joins together the nodes, taking into account the probability to belong to a cluster and the probability to join two different nodes. This aspects is useful in the definition of the cluster based on the study of the relationship between observations. Moreover, the *Label_prop* has been chosen because it is supposed that the proximity matrix identifying the closeness of the observation can be is an important support support for the propagation of the labels inside the dataset. Consequently, the community detection model can be able to identify clusters with a similar value for the outcome variable.
Other algorithms, as for instance *Cluster Spinglass* [226, 251], *Cluster Leading Eigen* [224] and *Cluster Edge Betweenness* [226] are not chosen for different aspects. Specifically, the *Cluster Spinglass* involves attributing a weight at each observation in the definition of community. On the contrary, the proposed algorithm takes into account the weights only in the first part in the definition of proximity matrix, and not in the second phase when the community detection algorithm are implemented to identify the cluster. For this reason, the above mentioned algorithm can not be included on CTST. The *Cluster Leading Eigen*

is based on a matrix defined as difference between the adjacency matrix and the probability that certain edges are present. Moreover, the *Cluster Edge Betweenness* is based on the *edge betweenness*, the ratio of the number of the shortest paths going through the edge to the shortest paths of all node pairs [315]. However, the paths are not considered in the CTSC, but the attention is posed on the presence or not of links and for this reason it is not included in CTSC.

After the application of the community detection algorithms, two different kinds of test are realized to define the best partition among the results obtained in the different iterations: the Tukey Honestly Significant Difference (HSD), used when the outcome variable is quantitative, and the Fisher test, used for qualitative outcome variable.

The Tukey Honestly Significant Difference (HSD) test is applied inside the algorithm in order to define the differences between groups in terms of average value of the outcome variable. The test, introduced by John Tukey (1949), operates defining the *honestly significant difference* value and comparing it with the difference between the average value of the groups. The *honestly significant difference* value is calculated as

$$HSD = Q_\alpha(s, n-s)s_{\bar{d}}$$

where $Q_\alpha(s, n-s)$ is called *Studentized rank value* and $s_{\bar{d}}$ is equal to:

$$s_{\bar{d}} = \sqrt{\frac{MS_{error}}{r}}$$

where $\alpha$ defines the significance level, $s$ represents the number of the average values to compare; and $r$ indicates the number of observations. The test involves comparing all differences between the average values $|y_{i.} - y_{j.}|$ with the $HSD$ and, if $|y_{i.} - y_{j.}| > HSD$, it is possible to state that the difference between $_i$ and $_j$ is significant.

The Tukey HSD is useful to find out the differences between the groups, taking into account the average value of the outcome variable.

The Fisher test is used for qualitative variables. It was proposed in the 1930s by Fisher. It is based on the null hypothesis that it is not possible to identify association between the rows and columns of the $n \times n$ table, as the example in Table 4.1. Established that the columns represent the study group and the rows represent the outcome, the null hypothesis explain the probability of having a particular outcome not being influenced by the study group, and the test evaluates whether the two study groups differ in the proportions with each outcome. The probability is calculated as

$$p = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!}$$

where $a, b, c, d$ are the individual frequencies of the $2 \times 2$ contingency table, and $n$ is the total frequency. Later, the probability is used to compute Fisher test p-value in order to reject the null hypothesis.
The Fisher is useful to find out the differences between the groups where the outcome variable is qualitative and it is not possible to apply a test based on the mean values.

| | Column 1 | Column 2 | Total |
|---|---|---|---|
| Row 1 | a | b | a+b |
| Row 2 | c | d | c+d |
| Total | a+c | b+d | a+b+c+d |

Table 4.1: General form for Fisher Test

## 4.3 Community Detection Tree-Based Algorithm for Semi-supervised Clustering (CTSC)

It will be now proposed an iterative approach of semi-supervised clustering. This approach has been defined as a *cluster associated with an outcome variable*, classification identified by Bair (2013), because the partition of the data is realized taking into account a specific variable, i.e. the outcome variable.
The algorithm is built combining tree based and community detection algorithms. Its aim is to define an algorithm able to identify a partition in which observations with a similar value of the outcome variable are joined together in the same group.
CTSC works in three step. The first step is called *pre-training phase*. Some elements are taken into account in this phase, specifically:

- the classifier $f_y$;
- the vector of outcome variable, $y$;
- the column vector of $n$ elements, $\mathbf{Z} = [\mathbf{z}_{(1)}, \mathbf{z}_{(2)}, \dots, \mathbf{z}_{(p)}]$;
- the number of iterations, $B$;
- the number of iteration in the pre-training phase, $B_p$, where $p$ is the number of covariates. It is established that $B \gg p$.

A classifier $f$ is trained taking into account: $y$, the vector of labels of $n$ elements that identifies the outcome variable, and $z$, the column vector of $n$ elements that identifies the covariates $x_j$. The classifier is defined through the tree-based algorithm. This phase is iterative. The tree is trained $B_p$ times. Initially, one covariate is chosen at a time and a weight $w$, equal to $\mathbf{w} = (1/p, 1/p, \ldots, 1/p)$, is attributed to each covariate.

Notation

- $\mathbf{Z} = [\mathbf{z}_{(1)}, \mathbf{z}_{(2)}, \ldots, \mathbf{z}_{(p)}]$ a data frame of column vectors $\mathbf{z}_{(j)}$ of $n$ elements
- $\mathbf{y}$ a vector of labels of $n$ elements
- $B \gg p$ number of iterations
- $\mathbf{1}$ as a column vector of $p$ ones

Initialization

- $\mathbf{w} = (1/p, 1/p, \ldots, 1/p)$ a vector of weights of $p$ elements
- $\mathbf{v} = (v_i = 0)$ with $i = 1, \ldots, p$
- $\mathbf{M}_{nn} = \mathbf{S}_{nn} = \mathbf{Q}_{nn} = \mathbf{0}$

**Algorithm 1** Pre-training:

```
1: for $k = 1$ to $p$ do
2:     $\hat{\mathbf{y}}^{(k)} = \hat{f}(\mathbf{y}, \mathbf{z}_{(k)})$
3:     $v_k = \hat{\Psi}[\mathbf{z}_{(k)} | \hat{\mathbf{y}}^{(k)}]$
4:     $\mathbf{M}_{ij} = \begin{cases} 0 & \text{if } y_i^{(k)} \neq y_j^{(k)} \\ 1 & \text{if } y_i^{(k)} = y_j^{(k)} \end{cases} i, j = 1, \ldots, n \wedge i \neq j$
5:     $\mathbf{S}^{(k)} = \mathbf{S}^{(k-1)} + \mathbf{M}$
6: end for
7: $\mathbf{w}^p = \mathbf{v} / (\mathbf{1}^\intercal \mathbf{v})$
```

Box 4.1: Pre-training phase

A tree is built in each iteration. All terminal nodes of the different trees are analyzed in order to evaluate how many times one generic observation $i$ is classified with another generic observation $j$. Specifically: if $i$ and $j$ are in same node, $\mathbf{M}_{ij} = 1$, otherwise if $i$ and $j$ are not in same node, $\mathbf{M}_{ij} = 0$

$$\mathrm{M}_{ij} = \begin{cases} 0 & \text{if } y_i^{(k)} \neq y_j^{(k)} \\ 1 & \text{if } y_i^{(k)} = y_j^{(k)} \end{cases} i, j = 1, \ldots, n \wedge i \neq j$$

The matrix $\mathbf{M}_{nn}$ is estimated in each iteration. At the end of the pre-training phase, the matrices obtained by trees in $B_p$ are added together to create the $\mathbf{S}_{nn}$. The total amount of times that the observations are classified together in the same nodes is measured.

It is decided to call the $\mathbf{S}_{nn}$ as *proximity matrix*, because it is a square matrix in which the entry in cell $(i, j)$ is a measure of the similarity between the observations.

**Algorithm 2** Training:

```
1: for $h = p + 1$ to $B$ do
2:     sample $1 \leq \delta \leq p$ according to $\mathbf{w}^{(h-1)}$
3:     $\hat{\mathbf{y}}^{(h)} = \hat{f}(\mathbf{y}, \mathbf{z}_{(\delta)})$
4:     $v_{\delta}^{(h)} = v_{\delta}^{(h-1)} + \hat{\Psi}[\mathbf{z}_{(\delta)}|\hat{\mathbf{y}}^{(h)}]$
5:     $\mathbf{w}^{(h)} = \mathbf{v}^{(h)}/(\mathbf{1}^{\intercal}\mathbf{v}^{(h)})$
6:     $\mathbf{M}_{ij} = \begin{cases} 0 & \text{if } y_i^{(h)} \neq y_j^{(h)} \\ 1 & \text{if } y_i^{(h)} = y_j^{(h)} \end{cases} i, j = 1, \ldots, n \wedge i \neq j$
7:     $\mathbf{S}^{(h)} = \mathbf{S}^{(h-1)} + \mathbf{M}$
8: end for
```

- $\mathbf{u} = \{\forall u \in \mathbf{S} : \exists! u \wedge u_i \geqslant u_{i+1}\}$
- $\mathbf{D}$ a matrix of $(|\mathbf{u}| - 1)$ column vectors $\mathbf{d}_{(r)}$ of $n$ elements

**Algorithm 3** Clustering:

```
1: for $r = 1$ to $(|\mathbf{u}| - 1)$ do
2:     $\mathbf{Q}ij = \begin{cases} 0 & \text{if } \mathbf{S}ij \geqslant u_r \\ 1 & \text{if } \mathbf{S}_{ij} < u_r \end{cases}$
3:     Set $\mathbf{d}_{(r)}$ as the member detected of community($\mathbf{Q}$)
4: end for
5: $\mathbf{d}^* = \arg\min_{\mathbf{d} \in \mathbf{D}} [g(\mathbf{y}, \mathbf{d})]$
```

Box 4.2: CTSC Algorithm

The pre-training phase is fundamental for two reasons. Firstly, weights for the observations and for the variables are defined through the tree algorithm: the observations are weighted taking into account iteratively the heterogeneity of the group they belong to. The variables are weighted taking into account their variable importance, $v_k = \hat{\Psi}[\mathbf{z}_{(k)}|\hat{\mathbf{y}}^{(k)}]$. At the end of the pre-training phase, the weight of the variable is equal to: $\mathbf{w}^p = \mathbf{v}/(\mathbf{1}^{\intercal}\mathbf{v})$. Secondly, the application of the classifier allows to define the *proximity matrix*. The proximity matrix has an important role because it gives information about the closeness of the observations. Specifically, since $\mathbf{S}$ indicates how many times two observations are classified together, a high value inside $\mathbf{S}$ indicates that two observations have a similar value of covariates and outcome variable. This means that $\mathbf{S}$ defines the similarity, i.e. the proximity, between the observations. In other words, the matrix measures of the relation of closeness between the observations. The amount of times two observation are classified together pro-

vides a value of the strength of the relationship between observations. This means that $s_{ij}$ explains a grade of the vicinity level of two observations.
The pre-training phase stops after $B_p$ iterations. It is followed by a second phase called *training.* In this phase, the classifier is trained $B - p$ times. The classifier is trained taking into account one covariate at a time. It involves considering the weighted variables and the weighted observations as defined in the first phase. The results of the classifier in the different iterations are used to define the matrix **M**, and the matrix **S**. The final output of the training phase is the proximity matrix resulting by the sum of all $\mathbf{M}_{nn}$ defined iteratively. Finally, in the last phase, called *clustering*, **S** becomes the input matrix for the application of the community detection algorithm and the definition of clusters.
**S** is transformed into **Q**. Specifically, in each iteration the value of $s_{ij}$ are taken into account. In the first step, the value 1 is attributed at the highest value of $s_{ij}$, and the value zero is attributed to the other $s_{ij}$. Consequently, the matrix **Q** will assume only two value: one for the highest value of $s_{ij}$ and zero for the other value.
Later, on the second iteration, the first two highest values of $s_{ij}$ are taken into account and they assumes value one; on the contrary, the value zero is applied to the other $s_{ij}$. Iteration by iteration, all values of $s_{ij}$ are gradually included in **Q**. Step by step, this transformation allows to evidence the high level of similatity between observations and to create a link between them.
In fact, the proximity matrix **S** is transformed in the adjacency matrix **Q**, characterized by 0/1 value. **Q** identifies the links between observations. This transformation defines the input for the third phase of the algorithm and the application of community detection algorithms. The aim is to define the clusters taking into account only the strongest relationships. In fact, $Q_{ij}$ is used to define the strength of the relationship between two observations.
It is important to evidence that, in CTSC, the clustering problem is transformed in a community detection problem, following the suggestion of Arruda et al. (2012), de Oliveira et al. (2008), Granell et al (2011, 2012). The community detection algorithm is applied on $\mathbf{Q}_{nn}$ iteratively for $B$ times. Each iteration generates a partition with a specific number of clusters. For each cluster, if the outcome variable is quantitative, the average value of the variable outcome is calculated and the Tukey Honestly Significant Difference (HSD) test is applied in order to define the difference between the average values of the outcome variable for the identified clusters. On the contrary, if the outcome variable is qualitative, the Fisher test is applied to evidence differences inside the identified clusters.
At the end, it is chosen firstly the partition with the minimum proportion of *p-value* higher

than a threshold $\alpha = 0.01$ is chosen at the final partition $m^*$:

$$m^* = \min\left(1 - \frac{\textit{number of p-value} \leq \alpha}{\textit{total number of p-value}}\right)$$

that defines the partition with the lowest proportion of *p-value* lower than or equal to $\alpha$. $m^*$ is the best partition, that is partition with groups considered as more different with respect to the outcome variable. The threshold $\alpha$ is usually lower than 0.01 As previously seen, the CTSC can be applied taking into account different community detection and tree based algorithms. CART and GBM have been chosen as classifier and *Louvain*, *Walktrap* and *Label_prop* as cluster algorithms. Six different combinations can be defined:

- the *Louvain* with CART;
- the *Louvain* with GBM;
- the *Walktrap* with CART;
- the *Walktrap* with GBM;
- the *Label_prop* with CART;
- the *Label_prop* with GBM.

These combinations offer the possibility to take advantage of different algorithms and to adapt to different datasets.
The proximity matrix defined through CART and GBM can be an important input for the definition of clusters. Specifically, it is supposed that the proximity matrix representing the strength of the relationship inside the dataset can become a correct base to identify clusters through the measure of modularity and of the *Louvain* algorithm. In fact, the matrix is able to measure the strength of the links between observations and for this reason can support correctly the estimation of the modularity. Additionally, it is hypothesized that the proximity matrix, indicating how many times the observations are classified together, provides a measure of the possibility that two observations belong to the same cluster. For this reason, it is also useful to estimate the $r$ distance applied on *Walktrap* algorithm. Finally, the proximity matrix, which identifies the closeness between the observations, may be useful in the definition of the neighbor between observations and it can facilitate the identification of similar labels and their propagation. Consequently, it can be a correct base for the application of *Label_prop*.
To conclude, an algorithm is proposed combining two different methods to define a semi-supervised clustering approach. The final output of the algorithm is a certain/given number of clusters, that are different between them for the value of the response variable, but similar

inside them w.r.t. the outcome variable.
In the following, CTST will be applied to stress its ability to reach the prefixed aims.

## 4.4 Related algorithms in literature

The CTSC supports the identification of groups inside a dataset. As evidenced before, CTSC is a semi-supervised clustering method associated with an outcome variable. However, CTSC differs form three algorithms that have been classified and presented in Chapter 1. CTSC differs from the algorithms theorized by Bair and Tibshirani (2004) and Gaymor and Bair (2013) because CTSC operates taking into account all features within the dataset in the whole process. On the contrary, the above mentioned algorithms define a subset of features highly correlated with the response variable in the starting phase and involve using only them. The use of a reduced number of features can support the identification of the groups that are highly correlated with the outcome variable. However, in this way a partial analysis of the phenomena is offered, reducing the information that can support the study. At the same time, CTSC differs from the algorithm proposed by Koester et al (2010), because CTSC does not make assumptions about the form of the distribution, but it starts defining a function to explain the relationship between the outcome variable and the features.
Generally, the cluster algorithms associated with an outcome variable are theorized to solve problems in the medical domain and, specifically, to identify subgroup of cancers [24, 137, 171]. The CTSC has not been theorized taking into account only a specific area: it can be applied on different datasets and on different research domains.
Besides the semi-supervised clustering associated with an outcome variable, CTSC differs from the other algorithms analyzed in Chapter 2, that combine tree-based methods and cluster analysis. For instance, CTSC differs from the model proposed by Barros et al. (2012) called *Clustering for improving Decision-Tree Induction* (Clus-DTI). In fact, Clus-DTI uses the cluster to classify the data in sub-groups and, later, applies on them the tree method. Its goal is to solve a classification problem. Clus-DTI is totally different from CTSC, because the application of the two methodologies is inverted - the cluster analysis is applied before the tree - and the aim is different.
In addition, CTSC differs from the *Clustering Feature Decision Tree model* (CFDT) proposed by Xu et al. (2011). The CFDT starts applying a micro clustering algorithm to summarize the data and to improve the accuracy for the successive application of decision tree induction. Although they have in common the aim and the definition of a semi supervised method, in fact their frameworks are different.
CTSC differs also from the *CLustering based on decision Trees* (CLTree) proposed by Liu

et al. (2005). In fact, CLTree combines trees and cluster analysis: specifically, it uses the decision tree to partition the data space into clusters. It involves mainly two steps. Firstly, a modified decision tree algorithm is used to build a cluster tree with the aim to capture the natural distribution of the data without making any prior assumption. Later, an interactive pruning step is performed to simplify the tree and to find meaningful clusters. The CLTree is totally different from CTSC not only for the different aim, but also because the two methodologies are combined and applied in different ways.
Furthermore, the main difference between the CFDT and the other algorithms is based on the fact that, in order to define the clusters, the CTSC does not use traditional cluster algorithms, but the community detection algorithms. Therefore, CTSC provides an innovative approach in the definition of the clusters.
To conclude, CTSC has three main advantages. Specifically, the CTSC:

- can be implemented using quantitative and qualitative outcome variables, as well as quantitative and qualitative covariates. This aspect makes the algorithm an important tool for the analysis of different datasets and for the definition of clusters;
- uses all covariates included in the dataset, not only a subset, giving consequently the possibility to consider the whole dataset;
- combines different tree-based and community detection algorithms in a flexible manner. It is therefore more adaptable to the study of different data and phenomena.

## 4.5 Empirical evidence

CTSC will be now applied on different datasets to evaluate its performance in different situation.
Two steps will be followed. Firstly, CTSC will be applied on simulated datasets to comprehend the capacity of CTSC to identify clusters inside the dataset taking into account qualitative and quantitative outcome variables, different level of the perturbation and overlapping on the data, different number and kinds of covariates, iterations and noise variables. Moreover, the CTST is applied a small dataset, the UNESCO dataset, in order to fix specific aspects of the algorithm, as for instance $B = 100$, and to use only the CART as classifier. The aim is to comprehend the functionality of the algorithm on a real dataset and w.r.t qualitative and quantitative outcome variables.
Secondly, the algorithm will be applied on two bigger datasets: the Boston and the rent99 datasets. The number of iterations will be improved until $B = 500$ and all possible combinations of CTSC will be applied. This second phase aims to evaluate the feasibility of

the algorithm taking into account wide datasets and the different results deriving from the application of different combinations.
Finally, the results of CTSC applied on Boston and rent99 will be compared with those generated by the application of the traditional cluster methods. This comparison aims to evaluate the existence of differences among the traditional cluster algorithms and the CTSC. Two aspects will be taken into account:

- the group compositions, to comprehend possible overlaps among the traditional cluster algorithms and the CTSC;
- the percentage of partitions with a highest proportion of *p-values* obtained for the Tukey HSD test in each possible specifications of CTSC, and for different value of $B$. The goal here is to evaluate the capacity of the different algorithms to identify groups that vary significantly one from another.

### 4.5.1 Simulated data

The simulation data sets have been created in order to comprehend the capacity of CTSC to identify cluster similar w.r.t. the outcome variables in different situations.
Specific aspects are established for the creation of simulated data. First, it is decided to create datasets composed by one outcome variable, which can be qualitative or quantitative, and a number of covariates equal to two, four and six in order to define datasets with different sizes. Moreover, it is decided to define datasets only with qualitative covariates, only with quantitative covariates or with both kinds of variables. The different combinations of qualitative and qualitative variables allow to evaluate the real capacity of the CTSC to reach the prefixed aim in different situations.
Moreover, other elements are taken into account in the creation of simulated datasets. Firstly, for qualitative and quantitative variables, it is introduced an overlapping level on the observations. Two opposed levels of overlapping are fixed: low and high. Secondly, the number of groups inside the datasets has been fixed equal to two, four and six in order to identify different level of clustering of the data. Thirdly, the noise variables are introduced in the model with the aim to introduce uncontrolled factor that adds variation in the data and to define all possible real situations and to better evaluate the capacity of CTSC to work on different datasets. Also for this element, the number of noise variables introduced on the model is fixed firstly to zero, and later to the maximum number of the variables included in the model. Additionally, only for quantitative variables, the different level of perturbation are inserted. *Data perturbation is a data security technique that adds noise to databases allowing individual record confidentiality* [305, p. 14]. Two different level of

perturbation are included on the simulation: low and high perturbation.
Defined the dataset, the CTSC is applied combing the CART with the three community detection algorithms *Louvain*, *Walktrap* and *Label_prop*. It is fixed a number of iterations equal to 100 and 500 in order to evidence the existence of differences in terms of identified cluster increasing the number. Finally, in order to evaluate the capacity of CTSC to identify groups similar to the real hypothesized groups, it is calculate the Rand Index. The indicator assumes value zero when the partitions identified through the CTSC is totally different by the partition identified on the simulated data, and value one when the partition are identical.
The CTSC is applied firstly on dataset where the outcome variable is qualitative. Analyzing the result shown in Appendix A from Table A.4 to A.6 where the number of iterations is fixed equal to 100, and the Table 4.2 where the number of iterations is equal to 100 and 500, it is possible to state that results suggest that CTSC works better when the number of observation is low, the number of qualitative covariates is null, the observations are not overlapped and the number of noise variable is null or low.

Table 4.2: CTST results, Louvain $B = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|
| **0.0001** | **2** | **2** | **0** | **1.0** | **100** | **0.95** |
| 0.0001 | 2 | 2 | 0 | 0.5 | 100 | 0.00 |
| 0.0001 | 2 | 2 | 0 | 0.0 | 100 | 0.00 |
| **0.0001** | **2** | **2** | **2** | **1.0** | **100** | **0.95** |
| **0.0001** | **2** | **2** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **100** | **0.95** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **100** | **1.00** |
| 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| **0.0001** | **2** | **4** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **0** | **1.0** | **500** | **0.95** |
| 0.0001 | 2 | 2 | 0 | 0.5 | 500 | 0.00 |

Table 4.2: CTST results, Louvain $B = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---:|---:|---:|---:|---:|---:|---:|
| 0.0001 | 2 | 2 | 0 | 0.0 | 500 | 0.00 |
| **0.0001** | **2** | **2** | **2** | **1.0** | **500** | **0.95** |
| **0.0001** | **2** | **2** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **500** | **0.95** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **500** | **1.00** |
| 0.0001 | 2 | 4 | 0 | 1.0 | 500 | 0.00 |
| 0.0001 | 2 | 4 | 0 | 0.5 | 500 | 0.00 |
| **0.0001** | **2** | **4** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **500** | **1.00** |
| 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.57 |
| **0.05** | **2** | **2** | **0** | **0.0** | **100** | **0.92** |
| 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.48 |
| 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.57 |
| 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.22 |
| 0.05 | 2 | 2 | 4 | 1.0 | 100 | 0.48 |
| 0.05 | 2 | 2 | 4 | 0.5 | 100 | 0.26 |
| 0.05 | 2 | 2 | 4 | 0.0 | 100 | 0.12 |
| 0.05 | 2 | 4 | 0 | 1.0 | 100 | 0.18 |
| 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.37 |
| 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.84 |
| 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.88 |
| 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.77 |
| 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.77 |
| **0.05** | **2** | **4** | **4** | **1.0** | **100** | **0.95** |
| 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.86 |
| 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.64 |
| **0.05** | **2** | **6** | **0** | **1.0** | **100** | **0.92** |
| 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.84 |
| 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.82 |
| 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.84 |
| 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.82 |
| 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.77 |

Table 4.2: CTST results, Louvain $B = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|
| 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.82 |
| 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.82 |
| 0.05 | 2 | 2 | 0 | 1.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 0 | 0.5 | 500 | 0.57 |
| **0.05** | **2** | **2** | **0** | **0.0** | **500** | **0.92** |
| 0.05 | 2 | 2 | 2 | 1.0 | 500 | 0.48 |
| 0.05 | 2 | 2 | 2 | 0.5 | 500 | 0.57 |
| 0.05 | 2 | 2 | 2 | 0.0 | 500 | 0.22 |
| 0.05 | 2 | 2 | 4 | 1.0 | 500 | 0.24 |
| 0.05 | 2 | 2 | 4 | 0.5 | 500 | 0.22 |
| 0.05 | 2 | 2 | 4 | 0.0 | 500 | 0.26 |
| 0.05 | 2 | 4 | 0 | 1.0 | 500 | 0.66 |
| 0.05 | 2 | 4 | 0 | 0.5 | 500 | 0.84 |
| 0.05 | 2 | 4 | 0 | 0.0 | 500 | 0.70 |
| 0.05 | 2 | 4 | 2 | 1.0 | 500 | 0.45 |
| 0.05 | 2 | 4 | 2 | 0.5 | 500 | 0.76 |
| 0.05 | 2 | 4 | 2 | 0.0 | 500 | 0.68 |
| 0.05 | 2 | 4 | 4 | 1.0 | 500 | 0.45 |
| 0.05 | 2 | 4 | 4 | 0.5 | 500 | 0.86 |
| 0.05 | 2 | 4 | 4 | 0.0 | 500 | 0.63 |
| 0.05 | 2 | 6 | 0 | 1.0 | 500 | 0.80 |
| **0.05** | **2** | **6** | **0** | **0.5** | **500** | **0.92** |
| 0.05 | 2 | 6 | 0 | 0.0 | 500 | 0.82 |
| 0.05 | 2 | 6 | 2 | 1.0 | 500 | 0.75 |
| **0.05** | **2** | **6** | **2** | **0.5** | **500** | **0.92** |
| 0.05 | 2 | 6 | 2 | 0.0 | 500 | 0.82 |
| 0.05 | 2 | 6 | 4 | 1.0 | 500 | 0.77 |
| 0.05 | 2 | 6 | 4 | 0.5 | 500 | 0.75 |
| 0.05 | 2 | 6 | 4 | 0.0 | 500 | 0.82 |

Then, it is applied the CTSC on simulated dataset that includes quantitative outcome variable. Analyzing the results shown from Table A.4 to A.6 where the number of observation is fixed equal to $B = 100$, and the Table 4.3 where $B = 100$ and $B = 500$, it is possible to observe that the value of Rand index decreases when the overlapping value is high and when the number of noise variable increases. Moreover, the presence of qualitative covariates does not influence the value of Rand Index.
On the contrary, this value changes taking into account the level of perturbation. Specifically, each combination between covariates, noise variables and qualitative variables are calculated considering two different levels of perturbation, high and low. The results evidence that the level of perturbation has a direct impact on the value of Rand Index. Specifically, when the level of perturbation is low, the value of Rand index is higher. This aspect evidences a difficulty for the CTSC to identify correctly clusters when the level of perturbation on the data is high.

Table 4.3: CTST results quantitative outcome, Louvain $n = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|
| **0.0001** | **2** | **2** | **0** | **1.0** | **100** | **0.96** |
| **0.0001** | **2** | **2** | **0** | **1.0** | **100** | **0.96** |
| 0.0001 | 2 | 2 | 0 | 0.5 | 100 | 0.31 |
| **0.0001** | **2** | **2** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **1.0** | **100** | **0.96** |
| **0.0001** | **2** | **2** | **2** | **1.0** | **100** | **0.96** |
| 0.0001 | 2 | 2 | 2 | 0.5 | 100 | 0.31 |
| **0.0001** | **2** | **2** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **100** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **100** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **100** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.0** | **100** | **0.96** |
| **0.0001** | **2** | **4** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **100** | **0.92** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **100** | **1.00** |

Table 4.3: CTST results quantitative outcome, Louvain $n = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---:|---:|---:|---:|---:|---:|---:|
| **0.0001** | **2** | **6** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.5** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **100** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **100** | **1.00** |
| 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.38 |
| 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.55 |
| 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.42 |
| 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.57 |
| 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.20 |
| 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.48 |
| 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.57 |
| 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.55 |
| 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.46 |
| 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.64 |
| 0.05 | 2 | 2 | 4 | 1.0 | 100 | 0.20 |
| 0.05 | 2 | 2 | 4 | 1.0 | 100 | 0.48 |
| 0.05 | 2 | 2 | 4 | 0.5 | 100 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.5 | 100 | 0.42 |
| 0.05 | 2 | 2 | 4 | 0.0 | 100 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.0 | 100 | 0.74 |
| 0.05 | 2 | 4 | 0 | 1.0 | 100 | 0.31 |
| 0.05 | 2 | 4 | 0 | 1.0 | 100 | 0.84 |
| 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.54 |
| 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.54 |
| 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.60 |
| 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.88 |
| 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.57 |
| 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.35 |
| 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.30 |
| 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.74 |
| 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.48 |
| 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.81 |
| 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.51 |
| 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.38 |
| 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.43 |
| 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.60 |
| 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.54 |
| 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.81 |
| 0.05 | 2 | 6 | 0 | 1.0 | 100 | 0.77 |
| 0.05 | 2 | 6 | 0 | 1.0 | 100 | 0.81 |
| 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.43 |
| 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.88 |
| 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.74 |
| 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.62 |
| 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.76 |
| 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.57 |
| **0.05** | **2** | **6** | **2** | **0.5** | **100** | **0.96** |

Table 4.3: CTST results quantitative outcome, Louvain $n = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---:|---:|---:|---:|---:|---:|---:|
| 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.33 |
| 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.88 |
| 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.81 |
| 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.75 |
| 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.81 |
| 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.66 |
| 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.19 |
| **0.05** | **2** | **6** | **4** | **0.0** | **100** | **0.95** |
| **0.0001** | **2** | **2** | **0** | **1.0** | **500** | **0.96** |
| **0.0001** | **2** | **2** | **0** | **1.0** | **500** | **0.96** |
| 0.0001 | 2 | 2 | 0 | 0.5 | 500 | 0.31 |
| **0.0001** | **2** | **2** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **1.0** | **500** | **0.96** |
| **0.0001** | **2** | **2** | **2** | **1.0** | **500** | **0.96** |
| 0.0001 | 2 | 2 | 2 | 0.5 | 500 | 0.31 |
| **0.0001** | **2** | **2** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **500** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **1.0** | **500** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **500** | **0.96** |
| **0.0001** | **2** | **2** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **2** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **4** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **0** | **0.0** | **500** | **1.00** |

Table 4.3: CTST results quantitative outcome, Louvain $n = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---:|---:|---:|---:|---:|---:|---:|
| **0.0001** | **2** | **6** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **2** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **1.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.5** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **500** | **1.00** |
| **0.0001** | **2** | **6** | **4** | **0.0** | **500** | **1.00** |
| 0.05 | 2 | 2 | 0 | 1.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 0 | 1.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 0 | 0.5 | 500 | 0.38 |
| 0.05 | 2 | 2 | 0 | 0.5 | 500 | 0.55 |
| 0.05 | 2 | 2 | 0 | 0.0 | 500 | 0.42 |
| 0.05 | 2 | 2 | 0 | 0.0 | 500 | 0.57 |
| 0.05 | 2 | 2 | 2 | 1.0 | 500 | 0.20 |
| 0.05 | 2 | 2 | 2 | 1.0 | 500 | 0.48 |
| 0.05 | 2 | 2 | 2 | 0.5 | 500 | 0.57 |
| 0.05 | 2 | 2 | 2 | 0.5 | 500 | 0.55 |
| 0.05 | 2 | 2 | 2 | 0.0 | 500 | 0.46 |
| 0.05 | 2 | 2 | 2 | 0.0 | 500 | 0.64 |
| 0.05 | 2 | 2 | 4 | 1.0 | 500 | 0.20 |
| 0.05 | 2 | 2 | 4 | 1.0 | 500 | 0.48 |
| 0.05 | 2 | 2 | 4 | 0.5 | 500 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.5 | 500 | 0.61 |
| 0.05 | 2 | 2 | 4 | 0.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 0 | 0.5 | 500 | 0.38 |
| 0.05 | 2 | 2 | 0 | 0.5 | 500 | 0.55 |
| 0.05 | 2 | 2 | 0 | 0.0 | 500 | 0.42 |
| 0.05 | 2 | 2 | 0 | 0.0 | 500 | 0.57 |
| 0.05 | 2 | 2 | 2 | 1.0 | 500 | 0.20 |
| 0.05 | 2 | 2 | 2 | 1.0 | 500 | 0.48 |
| 0.05 | 2 | 2 | 2 | 0.5 | 500 | 0.57 |
| 0.05 | 2 | 2 | 2 | 0.5 | 500 | 0.55 |
| 0.05 | 2 | 2 | 2 | 0.0 | 500 | 0.46 |
| 0.05 | 2 | 2 | 2 | 0.0 | 500 | 0.64 |
| 0.05 | 2 | 2 | 4 | 1.0 | 500 | 0.20 |
| 0.05 | 2 | 2 | 4 | 1.0 | 500 | 0.48 |
| 0.05 | 2 | 2 | 4 | 0.5 | 500 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.5 | 500 | 0.61 |
| 0.05 | 2 | 2 | 4 | 0.0 | 500 | 0.60 |
| 0.05 | 2 | 2 | 4 | 0.0 | 500 | 0.61 |
| 0.05 | 2 | 4 | 0 | 1.0 | 500 | 0.31 |
| 0.05 | 2 | 4 | 0 | 1.0 | 500 | 0.84 |
| 0.05 | 2 | 4 | 0 | 0.5 | 500 | 0.54 |
| 0.05 | 2 | 4 | 0 | 0.5 | 500 | 0.54 |

Table 4.3: CTST results quantitative outcome, Louvain $n = 100, 500$

| overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|
| 0.05 | 2 | 4 | 0 | 0.0 | 500 | 0.60 |
| 0.05 | 2 | 4 | 0 | 0.0 | 500 | 0.88 |
| 0.05 | 2 | 4 | 2 | 1.0 | 500 | 0.51 |
| 0.05 | 2 | 4 | 2 | 1.0 | 500 | 0.74 |
| 0.05 | 2 | 4 | 2 | 0.5 | 500 | 0.30 |
| 0.05 | 2 | 4 | 2 | 0.5 | 500 | 0.74 |
| 0.05 | 2 | 4 | 2 | 0.0 | 500 | 0.48 |
| 0.05 | 2 | 4 | 2 | 0.0 | 500 | 0.88 |
| 0.05 | 2 | 4 | 4 | 1.0 | 500 | 0.70 |
| 0.05 | 2 | 4 | 4 | 1.0 | 500 | 0.74 |
| 0.05 | 2 | 4 | 4 | 0.5 | 500 | 0.46 |
| 0.05 | 2 | 4 | 4 | 0.5 | 500 | 0.74 |
| 0.05 | 2 | 4 | 4 | 0.0 | 500 | 0.48 |
| 0.05 | 2 | 4 | 4 | 0.0 | 500 | 0.81 |
| 0.05 | 2 | 6 | 0 | 1.0 | 500 | 0.77 |
| 0.05 | 2 | 6 | 0 | 1.0 | 500 | 0.75 |
| 0.05 | 2 | 6 | 0 | 0.5 | 500 | 0.51 |
| 0.05 | 2 | 6 | 0 | 0.5 | 500 | 0.88 |
| 0.05 | 2 | 6 | 0 | 0.0 | 500 | 0.81 |
| 0.05 | 2 | 6 | 0 | 0.0 | 500 | 0.62 |
| 0.05 | 2 | 6 | 2 | 1.0 | 500 | 0.77 |
| 0.05 | 2 | 6 | 2 | 1.0 | 500 | 0.76 |
| 0.05 | 2 | 6 | 2 | 0.5 | 500 | 0.64 |
| **0.05** | **2** | **6** | **2** | **0.5** | **500** | **0.96** |
| 0.05 | 2 | 6 | 2 | 0.0 | 500 | 0.67 |
| 0.05 | 2 | 6 | 2 | 0.0 | 500 | 0.66 |
| 0.05 | 2 | 6 | 4 | 1.0 | 500 | 0.77 |
| 0.05 | 2 | 6 | 4 | 1.0 | 500 | 0.76 |
| 0.05 | 2 | 6 | 4 | 0.5 | 500 | 0.51 |
| 0.05 | 2 | 6 | 4 | 0.5 | 500 | 0.84 |
| 0.05 | 2 | 6 | 4 | 0.0 | 500 | 0.64 |
| **0.05** | **2** | **6** | **4** | **0.0** | **500** | **0.90** |

To sum up, it is possible to state that the results evidence that the CTSC works better when:

- the outcome variable is quantitative;
- overlapping is equal to 0.001;
- the number of the noise variables are equal to 2.

Moreover, for quantitative outcome variable, the low perturbation level is another aspect that can influence the capacity of CTCS to identify clusters. Instead, for the qualitative variable, the aspect that influence the result is the number of covariates qualitative, a value equal zero is preferred in clustering phase.

The first results evidence the capacity of CTSC to reach the prefixed aim. In the next part of this chapter, the algorithm is applied on real dataset in order to evaluate the capacity to work on real dataset.

### 4.5.2 The Unesco website data

Unesco website Data is a dataset that includes information about Unesco Tourism Websites (UTWs). In particular, in 2016 it has been created an ad hoc database by manually collecting data from 142 websites dedicated to 137 cultural and natural heritages of UNESCO located in France (41 UTWs), Italy (52 UTWs) and Spain (49 UTWs). For each UTW, specific variables have been identified and classified into three main groups. The first one is composed by 57 binary variables (Tables from A.7 to A.9) that are equal to one if the specific element is included in a UTW, and to zero otherwise. The second group is composed by the indicators that evaluate the effectiveness of UTWs (Table A.10), defined in terms of online success (measured by Global Rank and Rank In) and capacity to attract and to engage online visitors (measured by Bounce Rate, Daily Pageviews per Visitors and Daily Time on Site). Thirdly, a third group of variables is considered, including 7 quantitative variables obtained from the first group (Table A.11). These variables have been built considering the seven main characteristics of the UTWs and counting, for each one, the number of related binary variables that are equal to one. Finally, three more variables are included in this dataset: the first defines the country where the Unesco site is located, the second identifies the type of Unesco site (cultural, natural or both), and the third the type of website (destination website or Unesco website). Before applying CTSC on the Unesco website data, it has been decided to fix $B = 100$ and to use only CART as classifier, with the aim to simplify the application and to better understand the implications and the results.
The variable *Age* has been chosen as the outcome variable. It has been supposed that the different life times of Unesco sites can influence the online communication strategies. Chosen covariates are the country, the type of Unesco site (Cultural, natural or both), the type of website, the Global Rank, the Daily Pageviews per Visitor and the variables on Table A.11. CTSC is applied with the aim to identify clusters with similar observation with respect to the age of sites.
The algorithm is applied in three steps. In the pre-training phase, the classifier is built iteratively and the number of iteration is equal to the number of covariates, $B = 13$. During this phase, an initial weight is attributed to the variables, $w = \frac{1}{13}$: this weight changes in time considering the importance of the covariates. These weights are later used in the training phase.

In the second phase, the training phase, a classifier is built many times $B = 100 - 13 = 87$. At the end of the two phases, 100 trees are obtained. Both in the pre-training and in the training phase, the nodes of the trees are analyzed and the amount of times that observations are classified together in the same node is counted. The identified amounts are later attributed to the proximity matrix.
The final clustering phase starts when the matrix becomes the input for cluster analysis. For $B = 100$ times, the three community detection algorithms are applied
The Louvain algorithm is the first and is repeated iteratively. The central plot in Figure **??** shows how an increase of iterations number determines a reduction of the number of identified groups. The right box in Figure **??** presents a proportion of *p-values* lowest than 0.01. The value is estimated for each partition. The value of the proportion of *p-value* decreases as well as the number of iterations increases.
The Figure **??** shows the best iteration among the $B = 100$. It is the $75th$ iteration. It has two groups with two different distributions of $y$. The number of groups has been chosen automatically taking into account the maximum improvement in modularity.

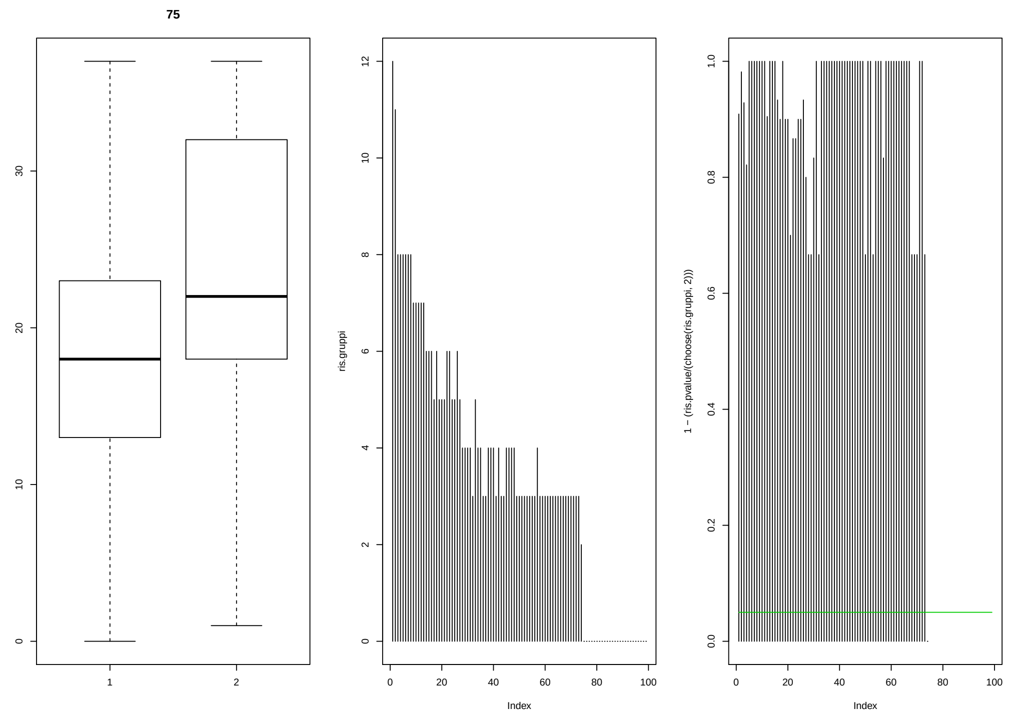
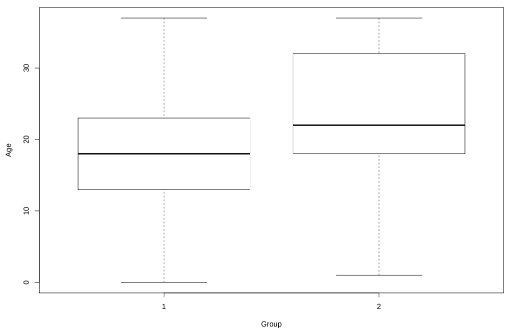


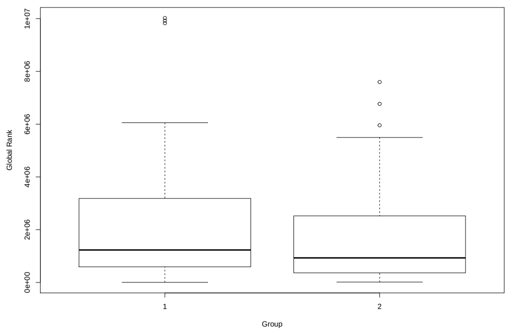
Figure 4.1: **Unesco Dataset, Louvain algorithm - 75th iteration**

In order to pinpoint the differences between two groups identified applying the *Louvain*, the box plots for the quantitative variable and the mosaic plots for the qualitative variable

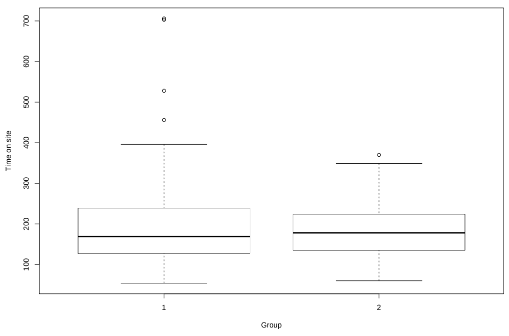
are realized. The box *a* of the Figure 4.2 shows the box plots of the response variable *Age* for the two different groups. The plot suggests that the groups are different and that they can be classified the first with the label "*the youngest*", the second with the label "*the oldest*". The analysis of the box plots of the quantitative covariates shows that generally the distributions of the covariates of *the youngest* websites are different compare to the distributions of *the oldest.* Specifically, the first groups has a greater variability w.r.t. the variables *global rank* (plot b Figura 4.2), *time on site* (plot *c* Figure 4.2), *daily page-views per visitor* (plot *d* Figure 4.2) and *brand* (plot *f* Figure 4.2). Moreover, different distributions for each group are identified for the covariate *contact and support* (plot *e* Figure 4.2), and for the variable *tourism* (plot *d* Figure 4.3), where *the oldest* group has higher values. This suggests the presence of information more in these websites w.r.t. the ones that belong to the the group *the youngest.* Instead, the two groups have similar distribution for the covariates *other* (plot *b* Figure 4.4) and *relational skills* (plot *c*Figure 4.4).
Then, considering the qualitative covariates (Figures 4.4) it is possible to state that a difference is recorded in the country composition of the groups. There are more French sites in the second group compared to the first. Moreover, it emerges that generally most Unesco site are cultural sites, and that hybrid site, i.e. cultural and natural, sites are equally distributed in the two groups. Finally, the number of Unesco sites that provide information through an owned website are equal in both groups (Website=yes).
Generally speaking, the results suggest that the oldest Unesco site obtain the best results both in terms of online success (they have a better Global rank), and in terms of elements included in their websites. On the contrary, the youngest Unesco sites have better results in terms of time spent in the website and of the number of visited pages. It is possible to state that the two groups identified in the partition are different not only for the outcome variable *Age*, but also for some above mentioned covariates.



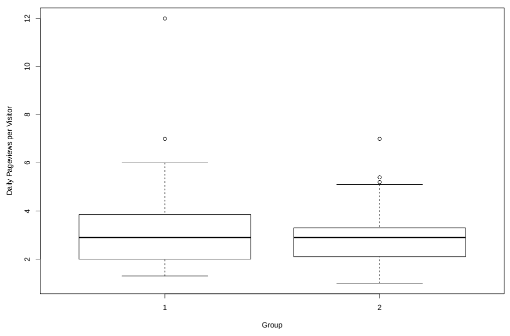
(a) *Age*



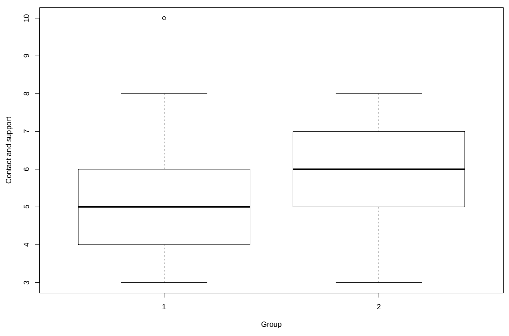
(b) *Global rank*



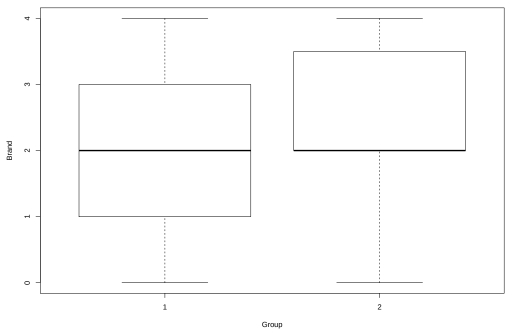
(c) *Time on site*



(d) *Daily pages per visitors*

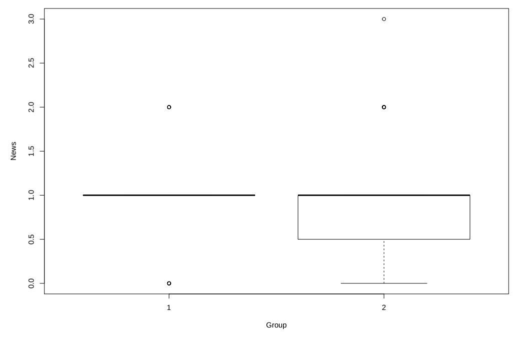


(e) *Contact and support*

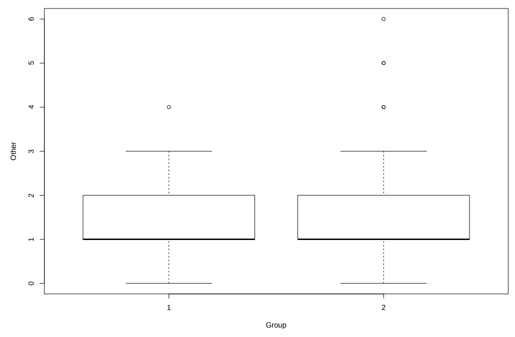


(f) *Brand*

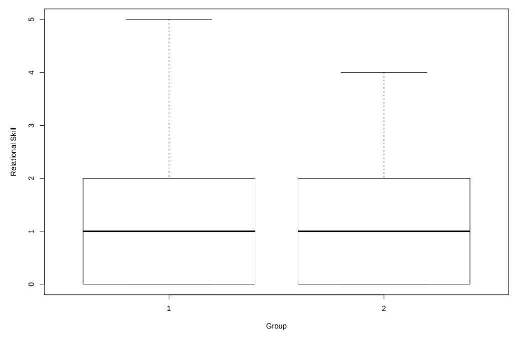
Figure 4.2: **Unesco data, Louvain, comparison between groups**



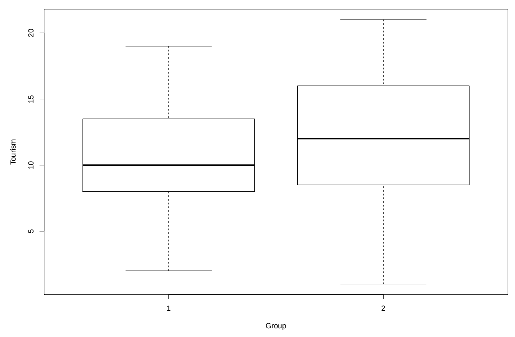
(a) *News*



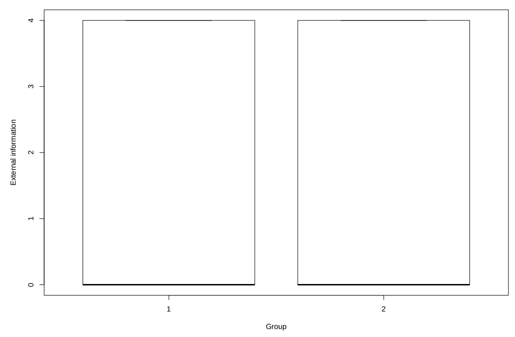
(b) *Other information*



(c) *Relational skills*

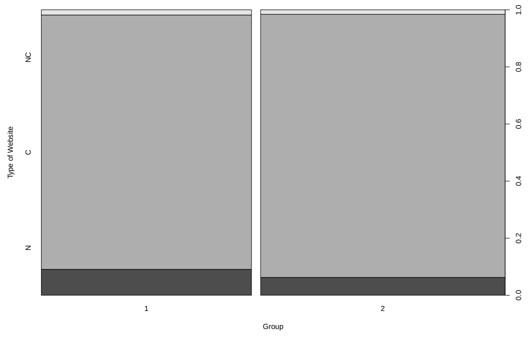


(d) *Tourism*

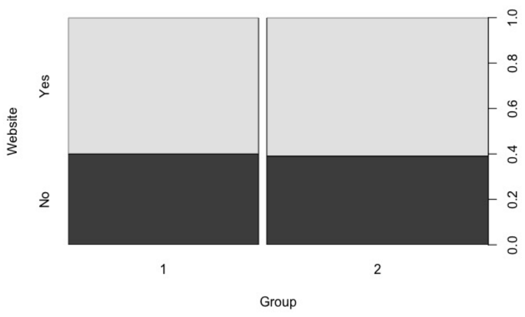


(e) *External information*

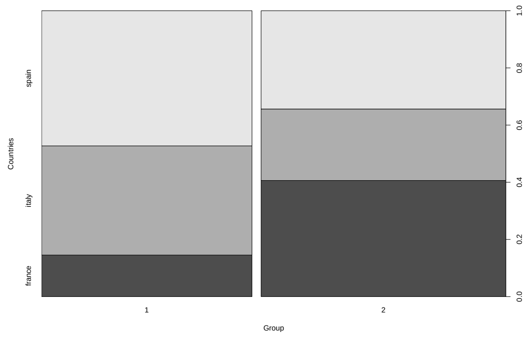
Figure 4.3: **Unesco data, Louvain, comparison between groups**



(a) *Type of Unesco site*



(b) *Website of Unesco site*



(c) *Countries*

Figure 4.4: **Unesco data, Louvain, comparison between groups**

The application of the second community detection algorithm, the *Walktrap*, gives different outcomes. Focusing on the cluster step of the algorithm, the results highlight how the number of groups tends to decrease when the iterations number increases, as shown in the central plot of Figure 4.5. However, this reduction is less steady than the results obtained using the *Louvain*. In addition, some peaks of the groups number are identified. These peaks correspond to an increase of the proportion of the *p-value* greater than 0.01. This bigger instability can be motivated by the fact that the number of iterations is not big enough to correctly estimate the $r$ distance, or by the size of the dataset that, as evidenced before, is not so relevant.
In the application of the *Walktrap*, the best iteration is the 29*th* (Figure 4.5). The left plot of Figure 4.5 shows how the two identified groups are totally different in terms of the distribution of the outcome variable. In fact, the two box plots are not overlapped.

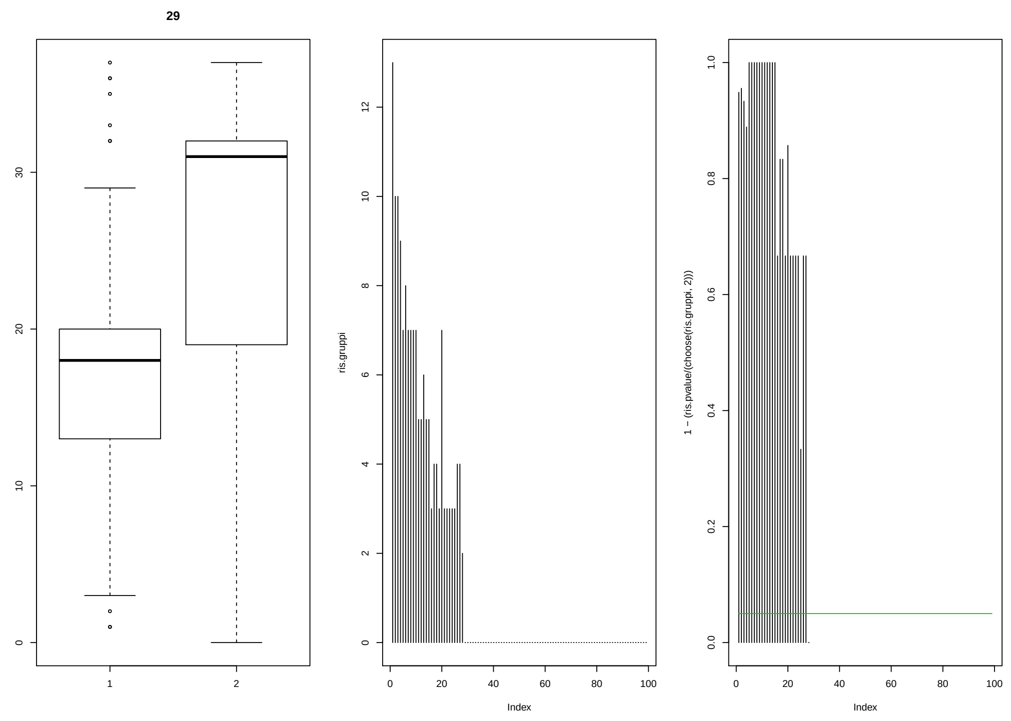
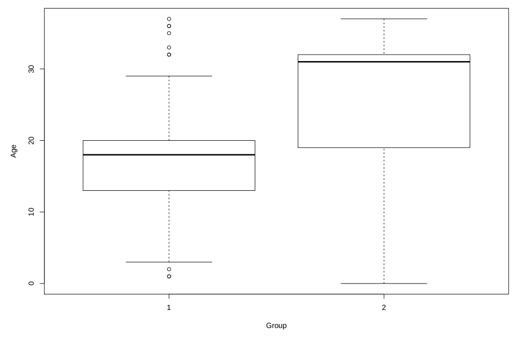


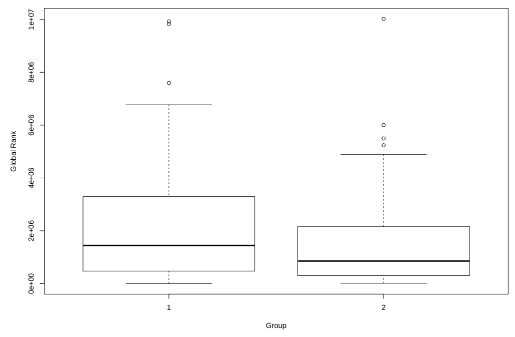
Figure 4.5: **Unesco Dataset, Walktrap algortihm - 29th iteration**

To better identify the differences between the first two groups, the distributions of the outcome variable and the covariates are analyzed. The two box plot in box *a* Figure 4.6 show how two totally different groups are identified taking into account the outcome variable *Age*. The first group includes *the youngest* Unesco sites, on the contrary *the oldest* are contained in the second group. As evidenced before, the two box plots are totally not overlapped.
The analysis of the covariates distributions highlights a greater variability for the first group. Specifically, a higher variability is recorded for the covariates: *global rank* (box *b* Figura 4.6), *brand* (box *f* Figure 4.6) and *relational skill* (box *c* Figure 4.7). Moreover, it is possible to identify higher values in the distribution of the *the oldest* for the covariates *time on site* (plot *c* Figure 4.6), *daily page per visitors* (plot *d* Figure 4.6).
The study of mosaic-plots of the qualitative covariates shows that the hybrid Unesco sites are included only in the second group (box *a* Figure 4.8). Moreover, the first group is composed mainly by the Italian Unesco sites, and the second by Spanish heritage sites (plot *c* Figure 4.8).
To sum up, the study of the covariates suggests that the oldest Unesco sites got the best results in terms of online success and information included in the websites. Additionally, the oldest sites have better results in terms of time spent on their websites. This result underlines how the *Walktrap* joins in the same groups the oldest sites together with the

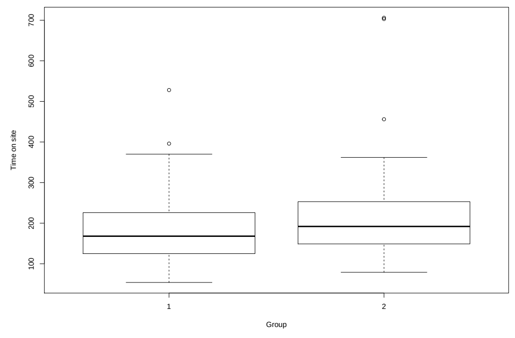
most successful for the online communication.



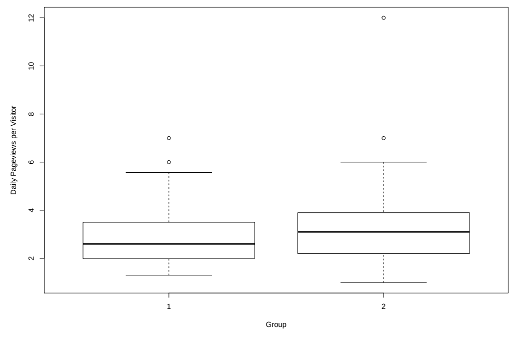
(a) *Age*



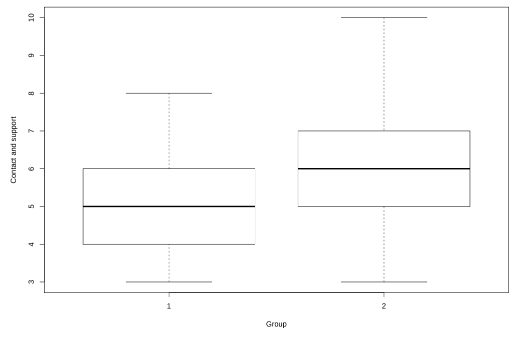
(b) *Global rank*



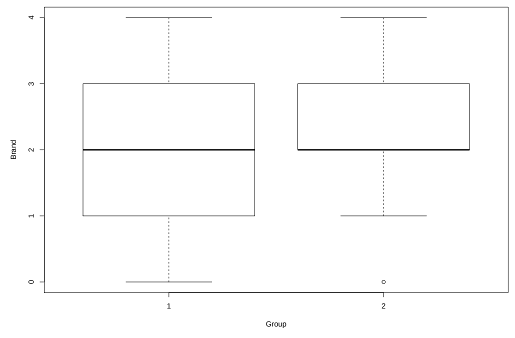
(c) *Time on site*



(d) *Daily pages per visitors*

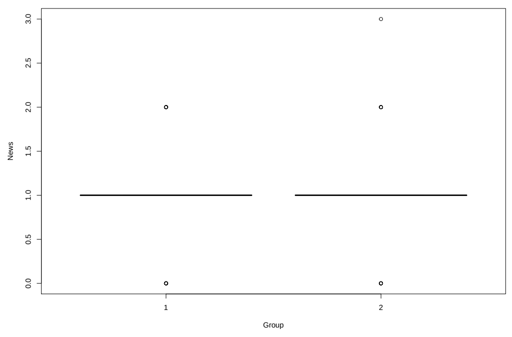


(e) *Contact and support*

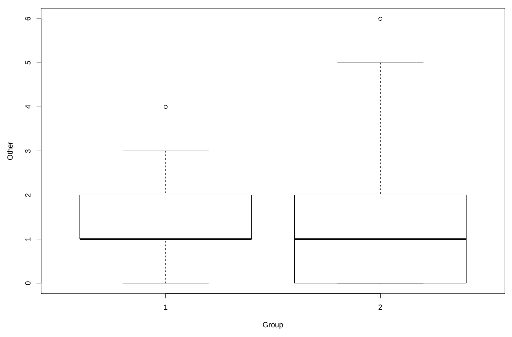


(f) *Brand*

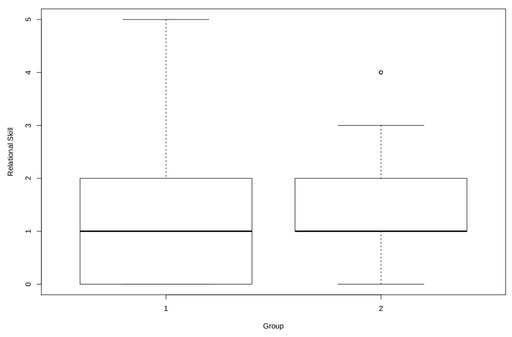
Figure 4.6: **Unesco data, Walktrap, comparison between groups**



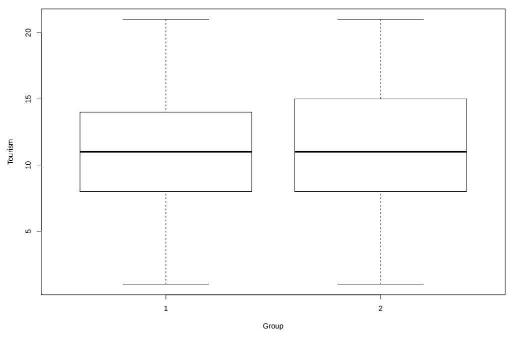
(a) *News*



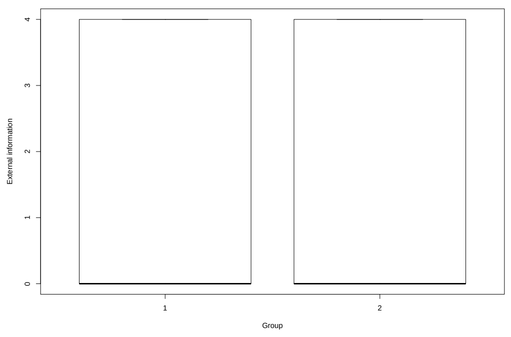
(b) *Other information*



(c) *Relational skills*

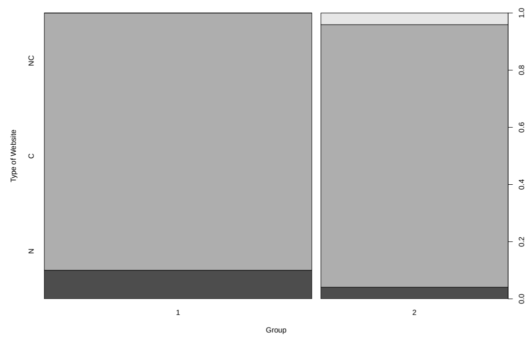


(d) *Tourism*

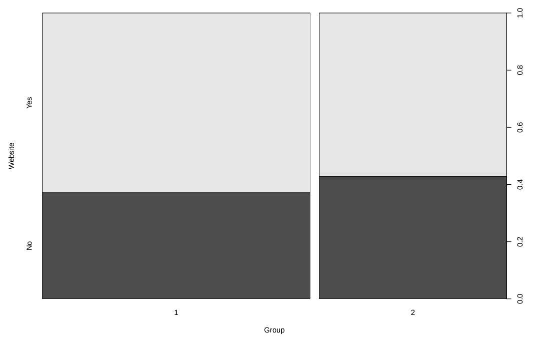


(e) *External information*

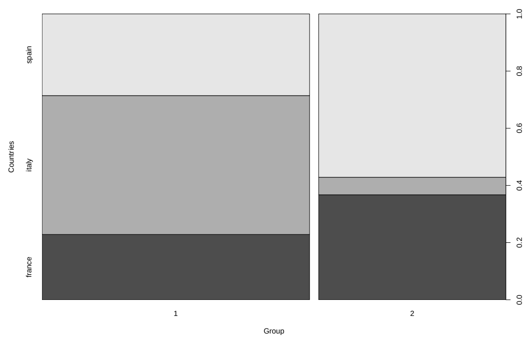
Figure 4.7: **Unesco data, Walktrap, comparison between groups**



(a) *Type of Unesco site*



(b) *Website of Unesco site*



(c) *Countries*

Figure 4.8: **Unesco data, Walktrap, comparison between groups**

Finally, the third community detection algorithm is applied: the *Label_prop.* The Figure 4.9 shows the partition with the lowest percentage of *p-value.* It is the number 23. In this case, too, the distributions of the outcome variable for the two groups in the best partition are different. The two box-plots are not overlapped.
As evidenced before in the application of *Louvain* and *Walktrap*, two different groups are identified. The *youngest* group has a bigger variability in some covariates, as for instance *global rank* (box *b* Figura 4.10). The oldest group has higher values in terms of *time on site* (plot *c* Figure 4.10). Finally, the first group is composed mainly by the Italian Unesco sites, and the second by Spanish heritage sites (plot *c* Figure 4.12).
One more time, the results highlight that two groups can be identified, the youngest and the oldest sites, and that the oldest Unesco sites are able to create websites with more information and to get the best results in terms of online success.

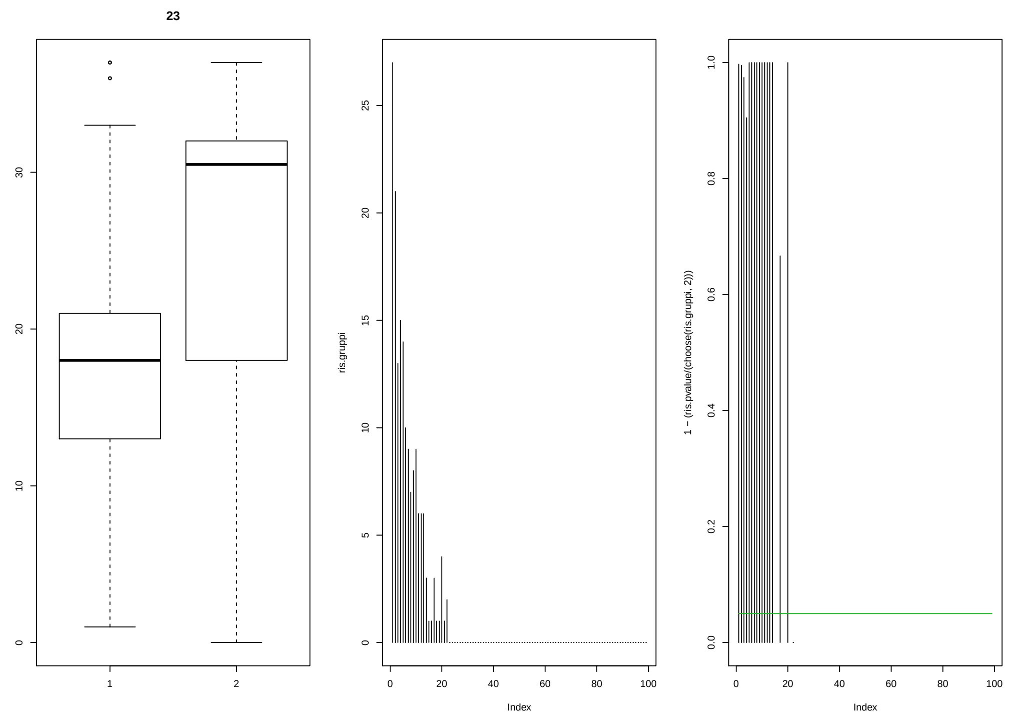
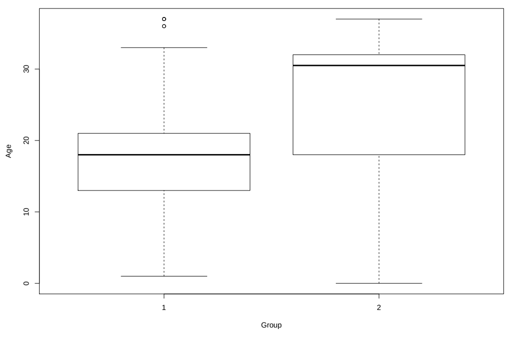
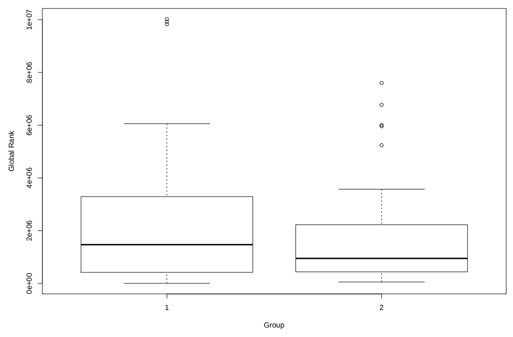


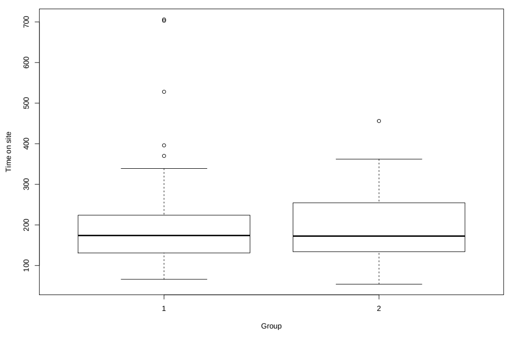
Figure 4.9: **Unesco Dataset, Label_prop algorithm - 23th iteration**



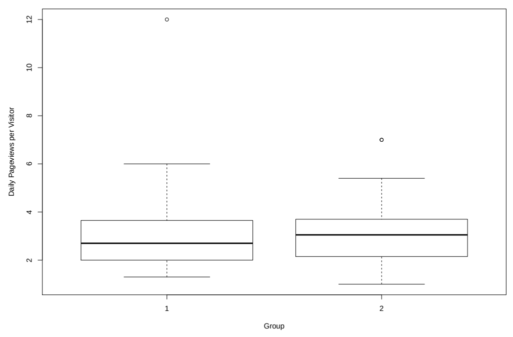
(a) *Age*



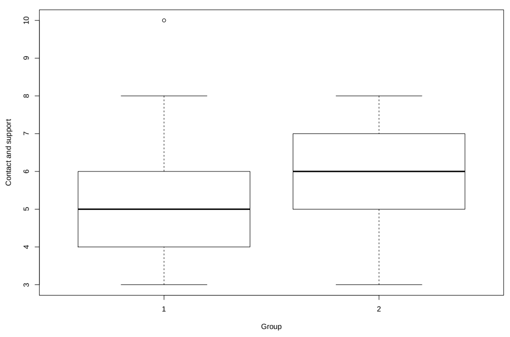
(b) *Global rank*



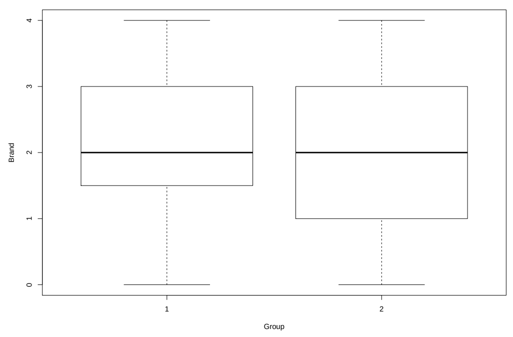
(c) *Time on site*



(d) *Daily pages per visitors*

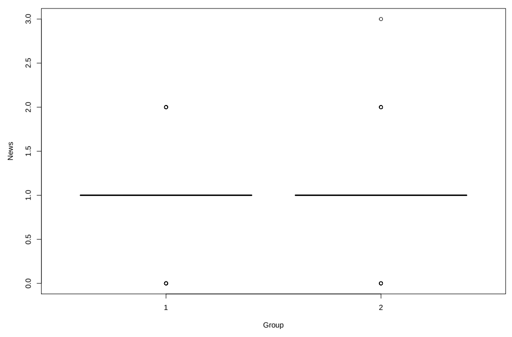


(e) *Contact and support*

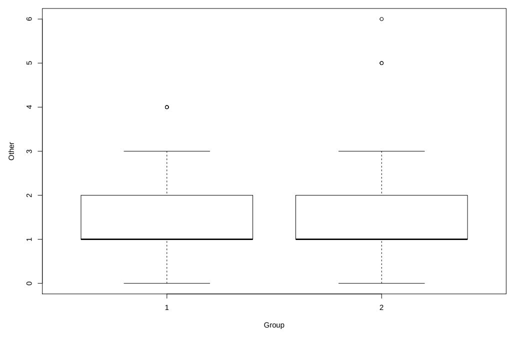


(f) *Brand*

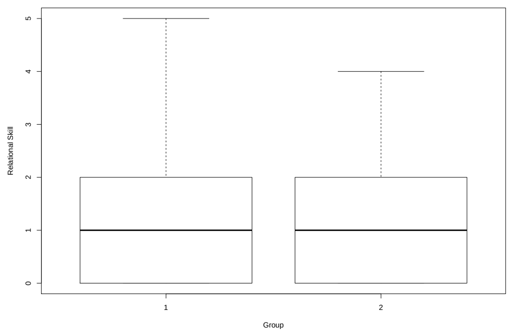
Figure 4.10: **Unesco data, Label_prop, comparison between groups**



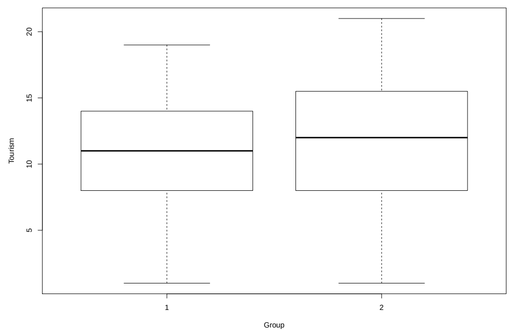
(a) *News*



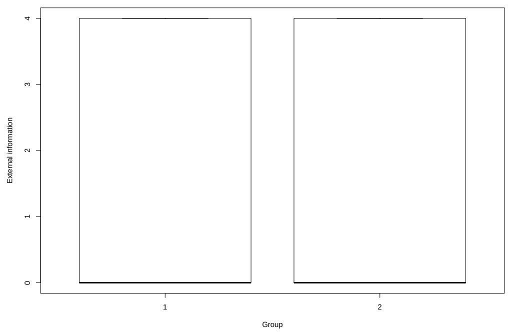
(b) *Other information*



(c) *Relational skills*

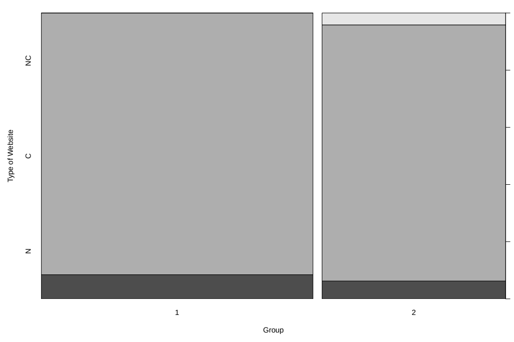


(d) *Tourism*

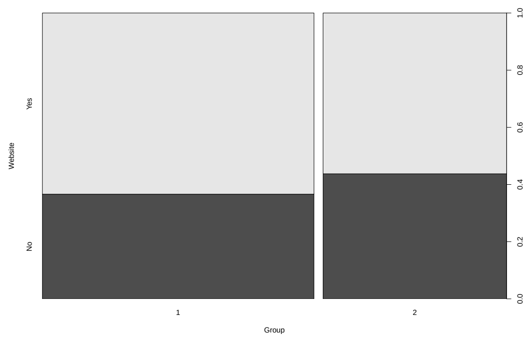


(e) *External information*

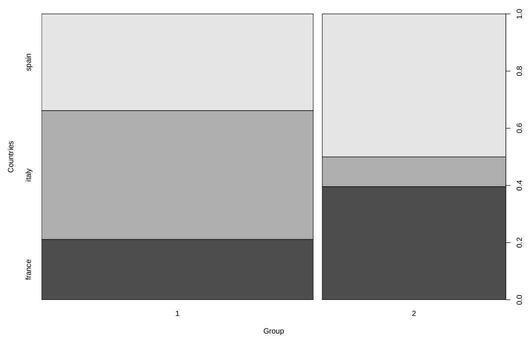
Figure 4.11: **Unesco data, Label_prop, comparison between groups**



(a) *Type of Unesco site*



(b) *Website of Unesco site*



(c) *Countries*

Figure 4.12: **Unesco data, Label_prop, comparison between groups**

Moreover, the composition of the different groups obtained applying *Louvain*, *Walktrap* and *Label_prop* is analyzed in order to underline differences. The Adjusted Random Index (ARI) is estimated. The indicator provides the opportunity to evaluate the groups composition of the different partitions and to compare it with the other cluster results. The Adjusted Rand Index (ARI) *is a measure of similarity, ranging from $c = 0$ when the two clusterings have no similarities, to $c = 1$ when the clusterings are in fact identical* [248, p. 847]. It is used to compare the partition pairs. It can be applied only when the number of groups is the same in the different compared partitions. The Table 4.4 shows as the partitions obtained through *Louvain* and *Walktrap* have a value of ARI close to 1.00: this means that the partitions have the same observations and the two community detection algorithms identify a similar result. The comparison between the *Label_prop* and *Louvain* and *Walktrap* defines a very low value of ARI: this means that the outputs of the algorithms are totally different.

| | Louvain | Walktrap | Label_prop |
|---|---|---|---|
| Louvain | 1.00 | 0.84 | 0.03 |
| Walktrap | 0.84 | 1.00 | 0.07 |
| Label_prop | 0.03 | 0.07 | 1.00 |

Table 4.4: The Adjusted Rand Index - Unesco data

The results of CTSC have been compared with the results of the traditional cluster methods. Specifically, the K-means and the Hierarchical clustering algorithms are applied on the Unesco dataset. For the k-means algorithm, it is chosen a value of $K = 2$ as the number of group obtained with the CTST (Figure 4.13). The attentions is posed on the outcome variable Age. Particularly, since the aim of CTSC is to define clusters similar w.r.t. the outcome variable, the boxplots of the outcome variable of the groups obtained by CTSC and the K-means are compared. The boxplots in Figure 4.14 evidences how the groups identified through the *Walktrap* and *Label_prop* present two distributions characterized by median values far more than the groups distributions of the outcome variable in *Louvain* and K-means. It means the *Walktrap* and *Label_prop* are more able to identify cluster dissimilar w.r.t. the outcome variable.
Moreover, if the CTSC results are compared with the hierarchical cluster and, in particular, with the Complete algorithm. It is realized the dendrogram labeling the observations w.r.t. CTSC cluster membership. As shown in Figure 4.15, the clusters identified by hierarchical clustering algorithm are not able to share the observations of the two groups as done by the CTSC. In fact, the observations classified in two different groups by CTSC are merged into different clusters. Moreover, the two groups obtained with Complete Linkage algorithm totally different for the size: the first includes 114 observations, the second only five observations. Additionally, the distribution of the outcome variable in the two groups obtained through *Complete Linkage* evidences a reduced differences between the median value of the two distributions, as shown in Figure 4.16.
To sum up, comparing this results of the two methods, it is possible to state that the CTSC is more able to identify groups inside the data and, specifically, to identify groups different for the value of the outcome variable. It is possible preliminary to state that the CTSC reach the prefixed aim operating with quantitative outcome variable.

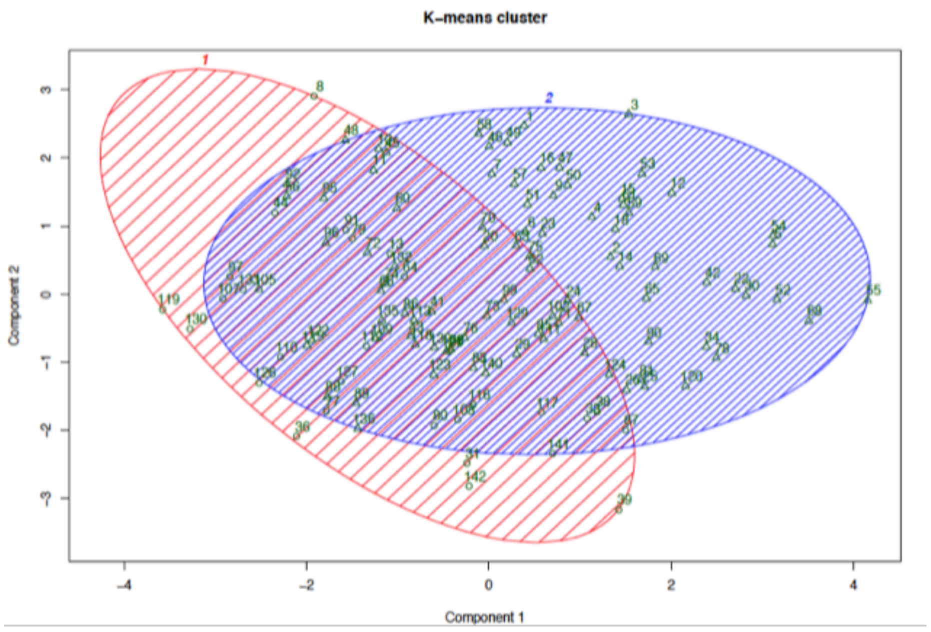
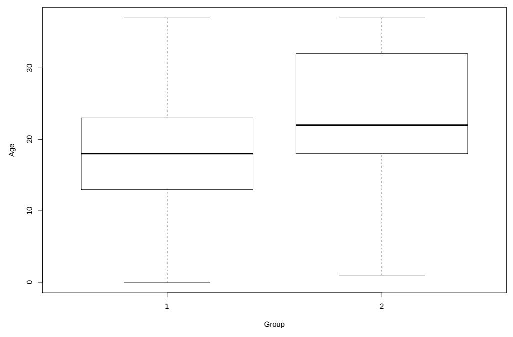
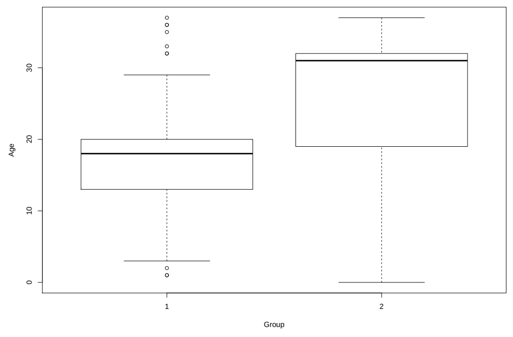


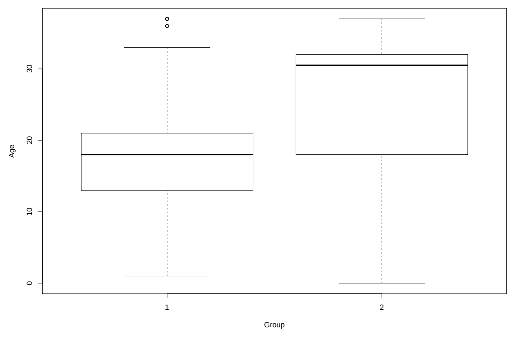
Figure 4.13: **Unesco data, K-means**



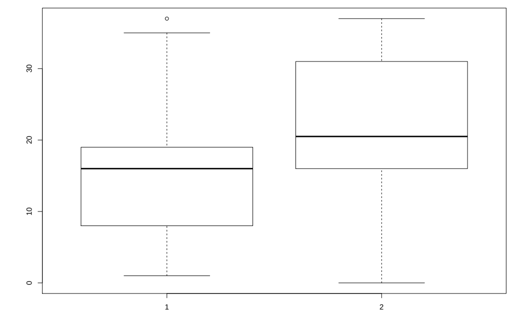
(a) *CTSC - Louvain*



(b) *CTSC - Walktrap*



(c) *CTSC - Label_prop*



(d) *K-means*

Figure 4.14: The boxplots of the outcome variable Age

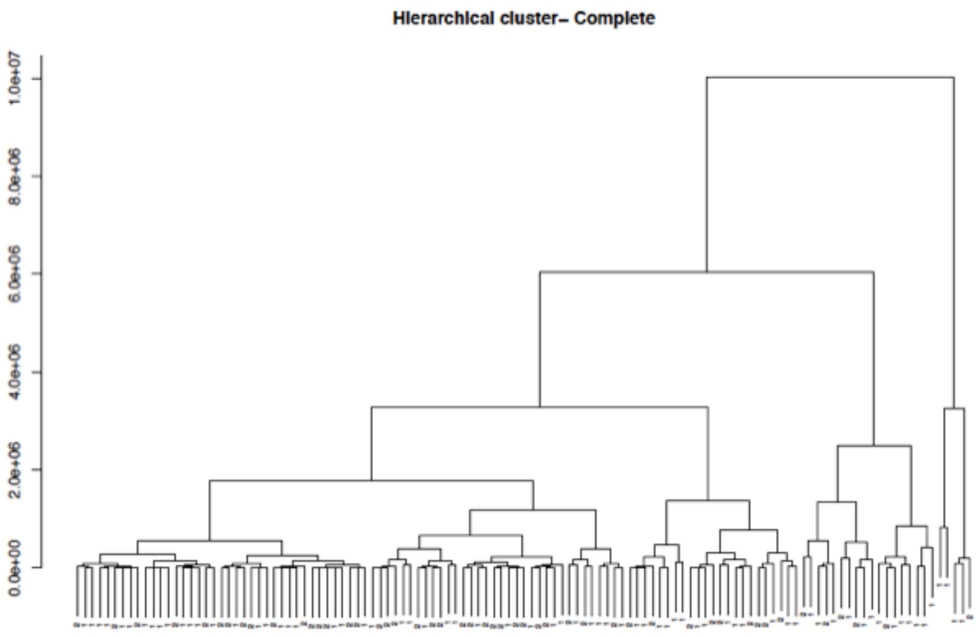
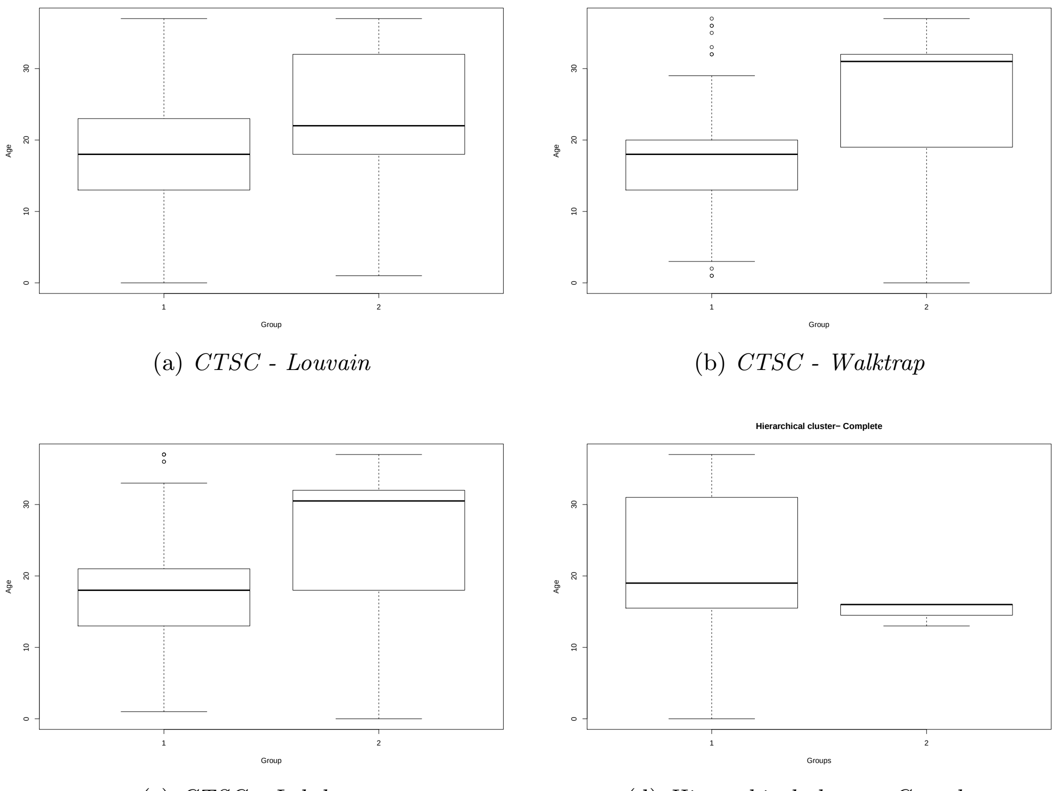


Figure 4.15: **Unesco data, Hierarchical cluster Complete**



(a) *CTSC - Louvain*

(b) *CTSC - Walktrap*

(c) *CTSC - Label_prop*

(d) *Hierarchical cluster- Complete*

Figure 4.16: The boxplots of the outcome variable Age

Then, a qualitative outcome variable is chosen to analyze the Unesco data. The variable *Countries* has been chosen to understand if there is a specificity related to the single countries that can influence the online communication strategy of the Unesco heritage sites. It is applied first the CTSC combining the CART with the three community detection algorithms. The application of *Louvain cluster* suggests the presence of three different clusters, as shown in Table 4.5. The results show how the first group is characterized for the presence of the heritage sites located in Italy, the second groups for the sites located in Spain, the third for the sites located mainly in France.
It is possible to state that the CTSC (*Louvain*) is able to identify groups characterized for a specific geographic area.

| | France | Italy | Spain |
|---|---|---|---|
| 1 | 3 | 19 | 5 |
| 2 | 4 | 13 | 30 |
| 3 | 27 | 5 | 13 |

Table 4.5: Qualitative outcome variable - CTSC "Louvain"

Later, it is applied the *Walktrap* on Unesco dataset. The application has allowed to identify two different groups as evidenced in Table 4.6. The first group includes heritage sites from Italy and Spain. The second is characterized for the presence mainly of the France Unesco sites.
In this case, it is not recorded a difference between the Countries, but the existence of a similarity between Spain and Italy.

| | France | Italy | Spain |
|---|---|---|---|
| 1 | 11 | 29 | 40 |
| 2 | 23 | 8 | 8 |

Table 4.6: Qualitative outcome variable - CTSC "Walktrap"

Finally, the application of the *Label_prop* community detection algorithm allow to identify three groups inside the Unesco dataset, as obtained before for the *Louvain* (Table 4.7). However, in this case, the first group includes roughly the same number of sites from the different countries. The second groups is characterized for the grater presence of the Spanish heritage sites. The last groups contains more Italian heritage site.
Also in this case, the difference between the geographic area is less evident that the results obtained through the *Louvain*.

|   | France | Italy | Spain |
|---|---|---|---|
| 1 | 17 | 17 | 14 |
| 2 | 15 | 1 | 25 |
| 3 | 3 | 19 | 9 |

Table 4.7: Qualitative outcome variable - CTSC "Label_prop"

To conclude, the CTSC reaches the goal of identifying different clusters taking into account a specific variable. The use of different community detection algorithms offers the possibility to evaluate the different results, and to evidence the differences inside the clusters taking into account the distribution of the outcome variables.
In the Unesco dataset, the results related to the quantitative outcome variable underline the presence of two specific groups. The first includes the youngest Unesco site and the second the oldest. The different combinations of algorithms shows how the oldest Unesco sites have been able to create successful websites, with more information than the youngest heritage sites. Moreover, the comparison with the results of traditional cluster algorithms evidence the capacity of CTSC to better identify cluster where observations within the same group are very similar to each other and different from observation in other groups. The same capacity is identified in the application of CTSC with a qualitative outcome variable. In the end, it is possible to state that the CTSC works well on small dataset, as it is able to identify groups similar to each other and different from observation in other groups.

### 4.5.3 Boston data

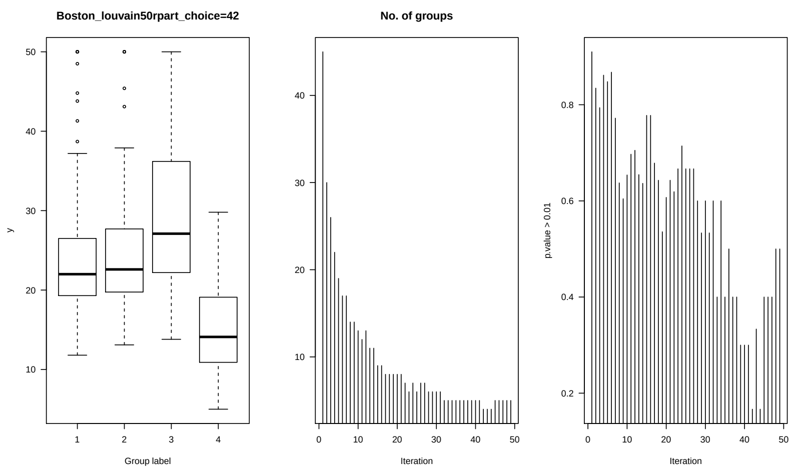
CTSC will be now applied on two different datasets in order to evaluate the usability and functionality. In this steps, the number of iterations $B$ will be increased from 50 to 500, and the GBM will be introduced as a second classifier in the pre-training and in the training phase.
The Boston data has been created by Harrison and Rubinfeld (1978) and has changed in time. Actually, it contains 506 rows and 20 columns. The 20 variables are:

- *town* is a factor with levels given by town names;
- *townno* is a numeric vector corresponding to the town;
- *tract* is a numeric vector of tract ID numbers;
- *lon* is a numeric vector of tract point longitudes in decimal degrees;
- *lat* is a numeric vector of tract point latitudes in decimal degrees;

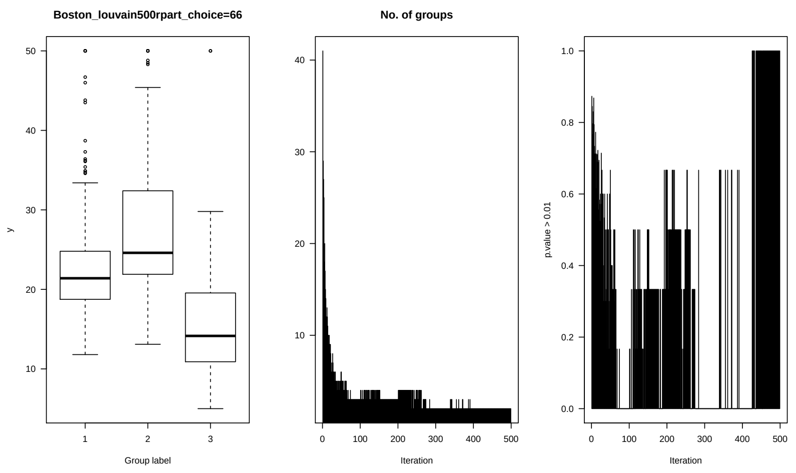
- *medv* is a numeric vector of median values of owner-occupied housing in USD 1000;
- *cmedv* is a numeric vector of corrected median values of owner-occupied housing in USD 1000;
- *crim* is a numeric vector of per capita crime;
- *zn* is a numeric vector of proportions of residential land zoned for lots over 25000 sq. ft per town;
- *indus* is a numeric vector of proportions of non-retail business acres per town;
- *chas* is a factor with levels 1 if tract borders Charles River; 0 otherwise;
- *nox* is a numeric vector of nitric oxides concentration (parts per 10 million) per town;
- *rm* is a numeric vector of average numbers of rooms per dwelling;
- *age* is a numeric vector of proportions of owner-occupied units built prior to 1940;
- *dis* is a numeric vector of weighted distances to five Boston employment centre;
- *rad* is a numeric vector of an index of accessibility to radial highways per town;
- *tax* is a numeric vector full-value property-tax rate per USD 10,000 per town;
- *ptratio* is a numeric vector of pupil-teacher ratios per town;
- *b* is a numeric vector of $1000 * (Bk - 0.63)^2$ where $B_k$ is the proportion of blacks;
- *lstat* is a numeric vector of percentage values of lower status population.

*Median value of owner-occupied homes in USD 1000's* has been chosen as outcome variable, the other 19 variables as covariates.
*Louvain cluster* is used as the first algorithm of community detection and is combined with CART and GBM (Appendix A - A.1 and A.2). $B$ is modified in time and assumes different values: $50, 100, 200, 300, 400, 500$. The first combination of CTSC with CART. Analyzing the left boxes in Figures A.1 and A.2, which shows the result of the best partitions, it is evident how the change of $B$ does not modify the number of identified groups in the best partition. For each iteration, the groups are three. The number of clusters has been determined automatically by the modularity.
Moreover, the study of the the central boxes shows how the increase of $B$ values determines a reduction and a major stability in terms of groups number. Finally, the analysis of the right plot, which represent the partitions with $p > 0.01$, shows how *p-value* decreases until

one hundred iterations, and then increases. The results suggests that CTSC is able to find groups inside the dataset with high difference between average values of the outcome also for lower value of $B$.
Secondly, GBM is used as classifier. Analyzing the left plot in Figure A.2, it is possible to state that firstly the distributions of the outcome variable are totally different comparing groups in each partition; secondly, the two groups identified in each partition are similar to the other two groups obtained in the other partitions. It means that the algorithm is generally able to identify groups with totally different *median value of owner-occupied homes in USD 1000's* and the increase of $B$ does not change significantly the composition of the groups. In fact, the comparison of the different box plots, and for the different iterations, shows how the increase of $B$ allows to obtain similar outcomes in terms of the number of groups and distribution of the outcome variable (left boxes in Figures 4.18).
In this case, too, the increase of the iterations number determines a decrease of the groups number (central boxes in Appendix A - Figure A.2 and 4.18). This decrease becomes steady when the number of iterations is high. In addition, the value of *p-value* reduces after the $300th$ iterations, as show in the right plot in Figure A.2.

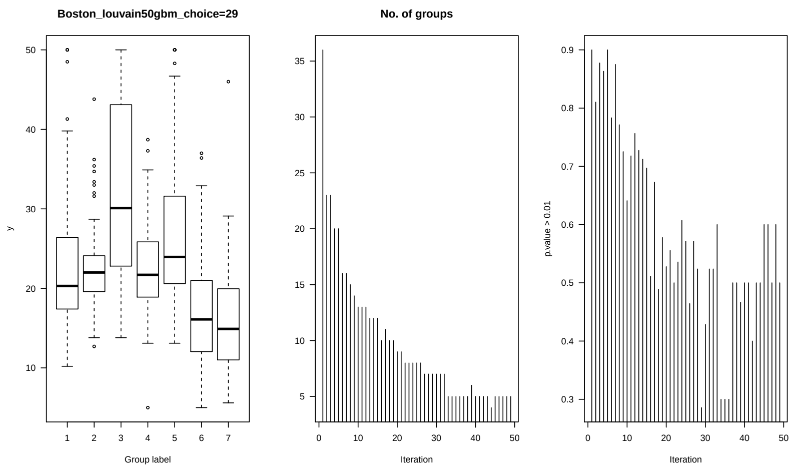


(a) *50 iteration*

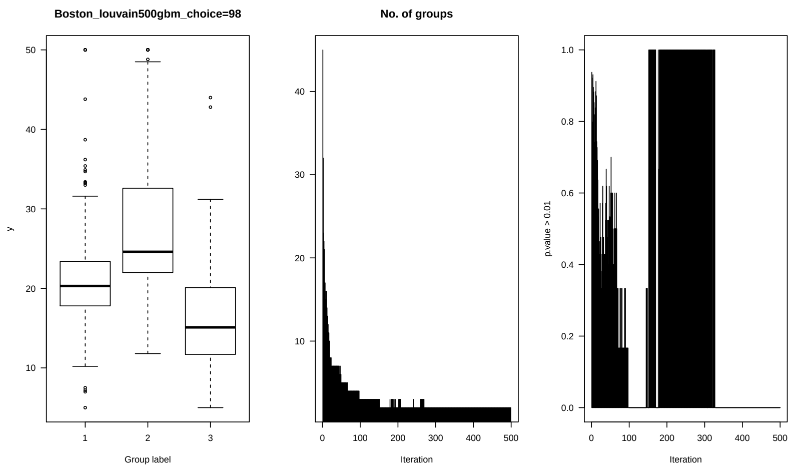


(b) *500 iteration*

Figure 4.17: **Algorithm applied including CART and Louvain - Best partitions**



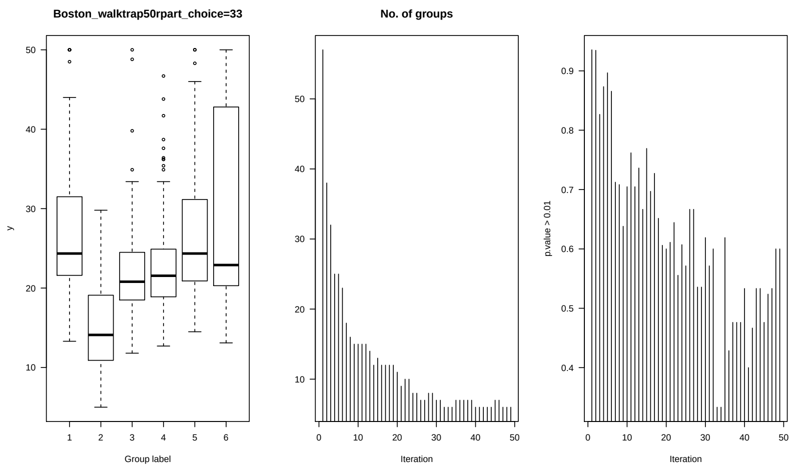
(a) *50 iteration*



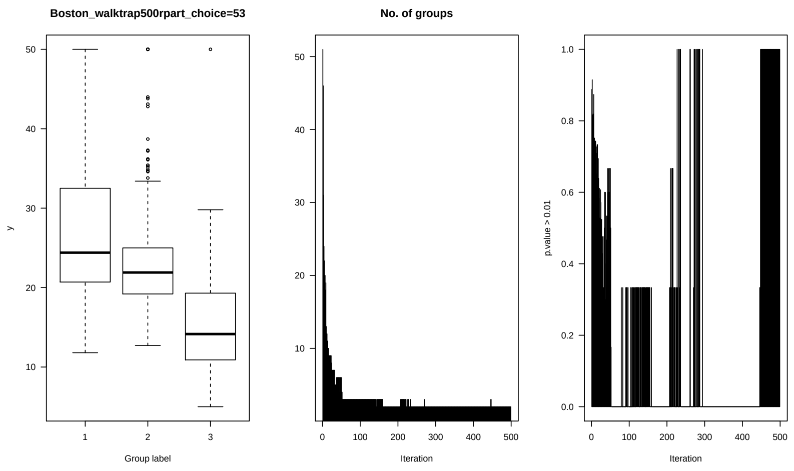
(b) *500 iteration*

Figure 4.18: **Algorithm applied including GBM and Louvain - Best partitions**

In a second step, CTSC is applied combining the *Walktrap* with CART and GBM algorithms (Appendix A - Figures from A.3 to A.4). The plots in Figures from A.3 to A.4 shows how the number of groups of the best partition is 3 (left box), the increase of $B$ determines a relevant decrease of the number of the groups (central box) and *p-value* generally decreases when the iterations number increases (right box).
As evidenced before for *Louvain*, it is possible to state that the algorithm identifies groups with different mean values of the response variable, and that the best partitions in the different iterations have overlapped box plots, showing a strong stability in the definition of the groups.

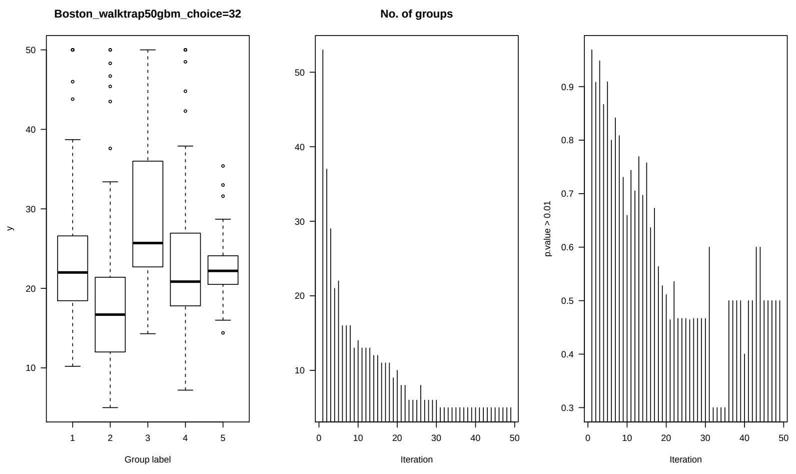


(a) *50 iteration*

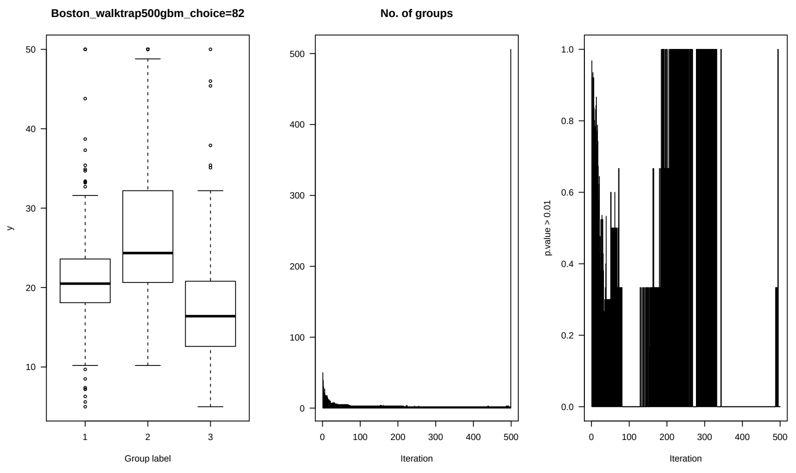


(b) *500 iteration*

Figure 4.19: **CTST applied on Boston data, CART and Walktrap - Best partitions**



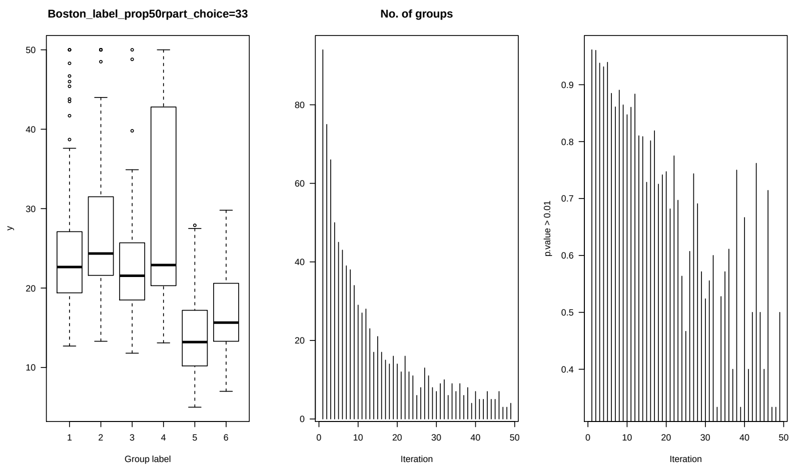
(a) *50 iteration*



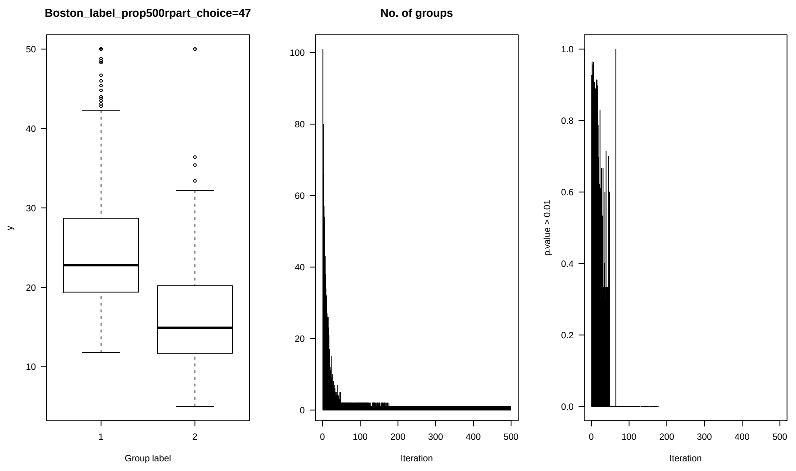
(b) *500 iteration*

Figure 4.20: **Algorithm applied including GBM and Walktrap - Best partitions**

Thirdly, the *Label Propagation* (*Label_prop*) community detection algorithm is applied (Appendix A - Figures A.5 to A.6) . As evidenced before for the other combinations, the groups number is equal to 2 or 3 in the best iterations (left box in Figures from A.5 to A.6), the number of the groups decreases quickly when the number of iterations increases (central box Figures from A.5 to A.6) and the partition with $p > 0.01$ decreases in time (right box in Figures from A.5 to A.6).
Also for this combination of CTSC, the goal of identifying groups that differ for the value of the outcome variable is reached (right box of Figures 4.21 and 4.22).

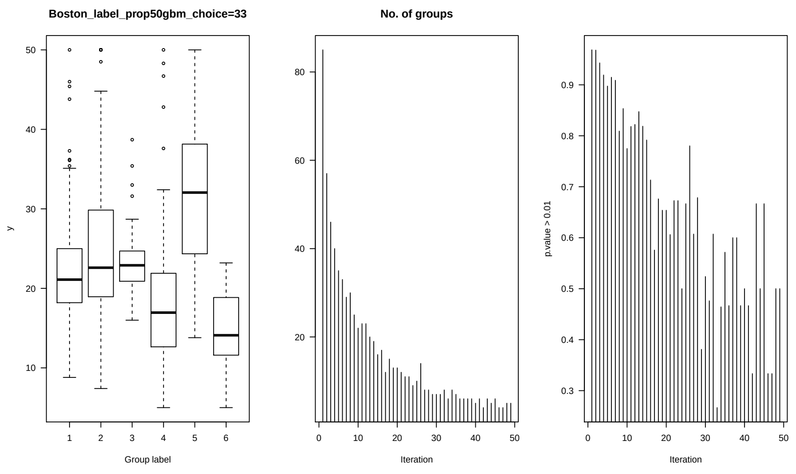


(a) *50 iteration*

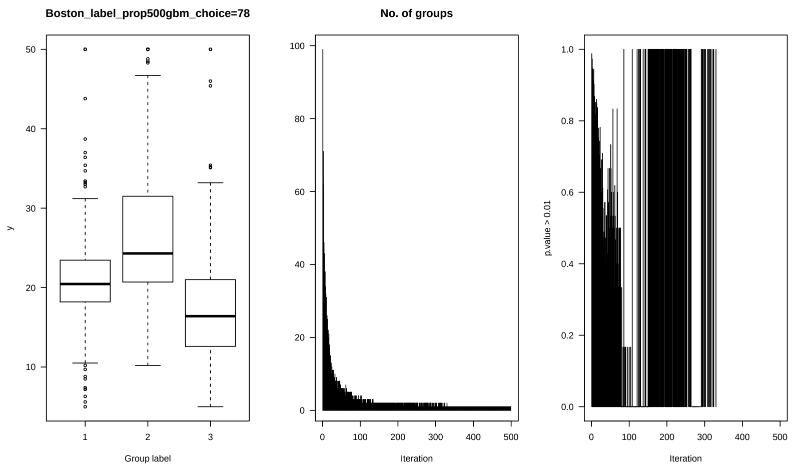


(b) *500 iteration*

Figure 4.21: **Algorithm applied including CART and Label Propagation - Best partitions**



(a) *50 iteration*



(b) *500 iteration*

Figure 4.22: **Algorithm applied including GBM and Label Propagation - Best partitions**

In order to evaluate the ability of the CTSC to identify groups with a significant difference between the average values, it is calculated the percentage of partitions with the lowest *p-value* for each combination of CTSC. The percentage of *p-value* from 0.05 to 0.0001 is taken into account. The same investigation has been used for the traditional clustering algorithms, with the aim of comparing the results and evaluating the existence of some differences.
The Table 4.8 shows the amount of partitions with the lowest *p-value*. Specifically, a value equal to 1 indicates that all partitions defined for a specific value of $B$ have a lower *p-value*, 100% of the partitions. On the contrary, a value of 0 indicates that all partitions have a lowest *p-value*. It emerges that the percentage increases when the value of $B$ decreases. Generally, CTSC has percentages equal to 100%. Only when $B = 50$ and considering $p < 0.01$,

the results are less than 0.50. This means that an high number of iterations allows to obtain groups that differ significantly for the differences between the mean. Moreover, it is very important to notice that some combinations of CTSC have a value equal to 1 for all values of *p-value*. The combinations of CART and *Louvain* and of GBM and *Label_prop* give the highest possible percentage from the 100*th* iteration. This shows how some combinations obtain good results even for low values of $B$.

The traditional clustering algorithms obtain similar results in terms of the number of identified groups and high percentage of partitions with low *p-value*. Only two algorithms, the *mcquitty* and the *median*, obtain a low percentage. It is also interesting to notice how the *single* is the only algorithm with the highest percentage for all values of *p-value*, but the result is not significant because it identifies only one group inside the dataset. All these results suggest that the CTSC has good performances in terms of clusters definition, taking into account the groups number and of the *p-value* results.

| Method | Groups | $p_{.05}$ | $p_{.01}$ | $p_{.001}$ | $p_{.0001}$ | $p_{.00001}$ | Average |
|---|---|---|---|---|---|---|---|
| walktrap50rpart | 6 | 0.67 | 0.67 | 0.53 | 0.33 | 0.33 | 0.51 |
| louvain50rpart | 4 | 0.83 | 0.83 | 0.83 | 0.67 | 0.67 | 0.77 |
| label prop50rpart | 6 | 0.73 | 0.67 | 0.53 | 0.47 | 0.47 | 0.57 |
| walktrap100rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain100rpart | 3 | **1.00** | **1.00** | **1.00** | **1.00** | 0.67 | 0.93 |
| label prop100rpart | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| walktrap200rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain200rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| label prop200rpart | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| walktrap300rpart | 3 | **1.00** | **1.00** | **1.00** | 0.67 | 0.67 | 0.87 |
| louvain300rpart | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| label prop300rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| walktrap400rpart | 3 | **1.00** | **1.00** | **1.00** | **1.00** | 0.67 | 0.93 |
| louvain400rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| label prop400rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| walktrap500rpart | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| louvain500rpart | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| label prop500rpart | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| walktrap50gbm | 5 | 0.70 | 0.70 | 0.60 | 0.40 | 0.30 | 0.54 |
| louvain50gbm | 5 | 0.70 | 0.70 | 0.50 | 0.50 | 0.50 | 0.58 |
| label prop50gbm | 5 | 0.70 | 0.70 | 0.70 | 0.70 | 0.60 | 0.68 |
| walktrap100gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain100gbm | 4 | 1.00 | 1.00 | 0.83 | 0.67 | 0.67 | 0.83 |
| label prop100gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| walktrap200gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain200gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| label prop200gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| walktrap300gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain300gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| label prop300gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| walktrap400gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| louvain400gbm | 3 | **1.00** | **1.00** | **1.00** | 1.00 | 0.67 | 0.93 |
| label prop400gbm | 3 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| walktrap500gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain500gbm | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| label prop500gbm | 3 | **1.00** | **1.00** | **1.00** | 0.67 | 0.67 | 0.87 |
| ward.D | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| ward.D2 | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| single | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| complete | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| average | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| mcquitty | 4 | 0.67 | 0.33 | 0.33 | 0.33 | 0.33 | 0.40 |
| median | 4 | 0.67 | 0.33 | 0.33 | 0.33 | 0.33 | 0.40 |
| centroid | 3 | **1.00** | 0.67 | 0.67 | 0.67 | 0.67 | 0.73 |
| kmeans | 4 | **1.00** | 0.83 | 0.67 | 0.50 | 0.50 | 0.70 |

Table 4.8: Application of different algorithm on Boston dataset

Moreover, Table 4.9 shows how the same number of groups are identified for the majority of CTSC combinations and for the traditional clustering algorithms. This aspect underlines the necessity to analyze the group composition to identify possible overlapping. The Adjusted Rand Index (ARI) is therefore estimated in order to verify the cluster composition of the groups obtained by CTSC compared with the cluster composition of the groups obtained by the traditional clustering methods. The indicator offers the opportunity to evaluate the groups composition of the different partitions and to compare it with the other cluster results. It is also used to compare the partitions pairs. Since ARI needs the same number of groups for the two partitions to be compared, solely the algorithms with three optimal groups were taken into account. From the analysis of the results in Table 4.9, it emerges how the traditional clustering methods produce partitions very similar to each other. For instance, ARI presents value equal to 0.9128576 comparing complete to either average or centroid, value equal to 1 comparing average to centroid. On the other hand, among semi-supervised method based on network *Walktrap* and *Label_prop* generated partitions similar with an ARI equal to 0.7780845, but more different to *Louvain* algorithm, having an ARI of 0.4787485 and 0.5014475, respectively. Moreover, extremely different there were the partitions between any traditional clustering method and any semi-supervised method based on networks. The most similar partitions were generated by *Walktrap* and complete with an ARI equal to 0.17568721, whereas the most different by *Label_prop* and both average and centroid with an ARI equal to 0.08023318.

| | walktrap | louvain | label_prop | ward.D | ward.D2 | complete | average | centroid |
|---:|---:|---:|---:|---:|---:|---:|---:|---:|
| walktrap | 1.00 | 0.48 | 0.78 | 0.16 | 0.16 | 0.18 | 0.15 | 0.15 |
| louvain | 0.48 | 1.00 | 0.50 | 0.09 | 0.09 | 0.10 | 0.08 | 0.08 |
| label_prop | 0.78 | 0.50 | 1.00 | 0.08 | 0.08 | 0.10 | 0.08 | 0.08 |
| ward.D | 0.16 | 0.09 | 0.08 | 1.00 | 1.00 | 0.94 | 0.97 | 0.97 |
| ward.D2 | 0.16 | 0.09 | 0.08 | 1.00 | 1.00 | 0.94 | 0.97 | 0.97 |
| complete | 0.18 | 0.10 | 0.10 | 0.94 | 0.94 | 1.00 | 0.91 | 0.91 |
| average | 0.15 | 0.08 | 0.08 | 0.97 | 0.97 | 0.91 | 1.00 | 1.00 |
| centroid | 0.15 | 0.08 | 0.08 | 0.97 | 0.97 | 0.91 | 1.00 | 1.00 |

Table 4.9: Adjusted Rand Index

Finally, the R packege Mclust is applied on Boston data. The package is used for model-based clustering, classification, and density estimation based on finite normal mixture modeling. It involves assuming a variety of data models and applying the maximum likelihood estimation and Bayes criteria in order to identify the most likely model and number of clusters. The application on Boston data is realized with the aim to compare the result of identification of the best partitions respect to the results obtained through the CTSC.

The results evidence the presence of six groups on the Boston data obtained by the VEV model, as show in Figure 4.23. The same number of clusters has been obtained through CTSC combined with CART and the two community detection algorithms *Walktrap* and *Label_prop* when the number of iterations has been fixed equal to 50.

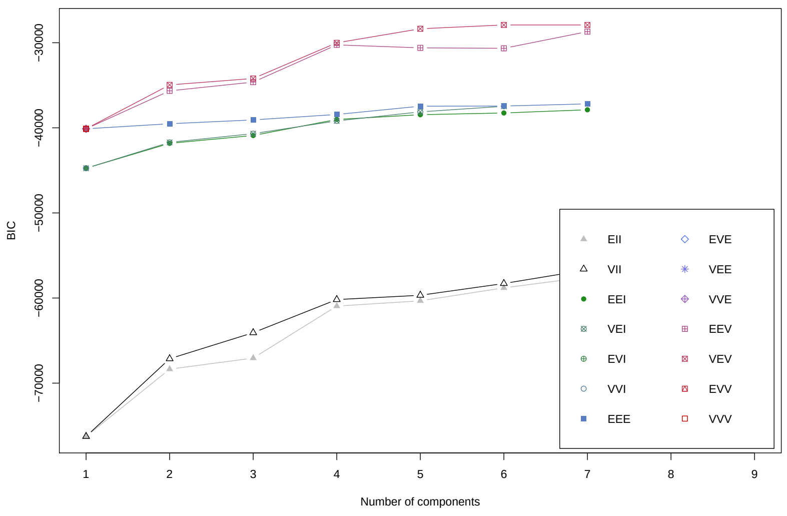


Figure 4.23: **Mclust Boston data- BIC**

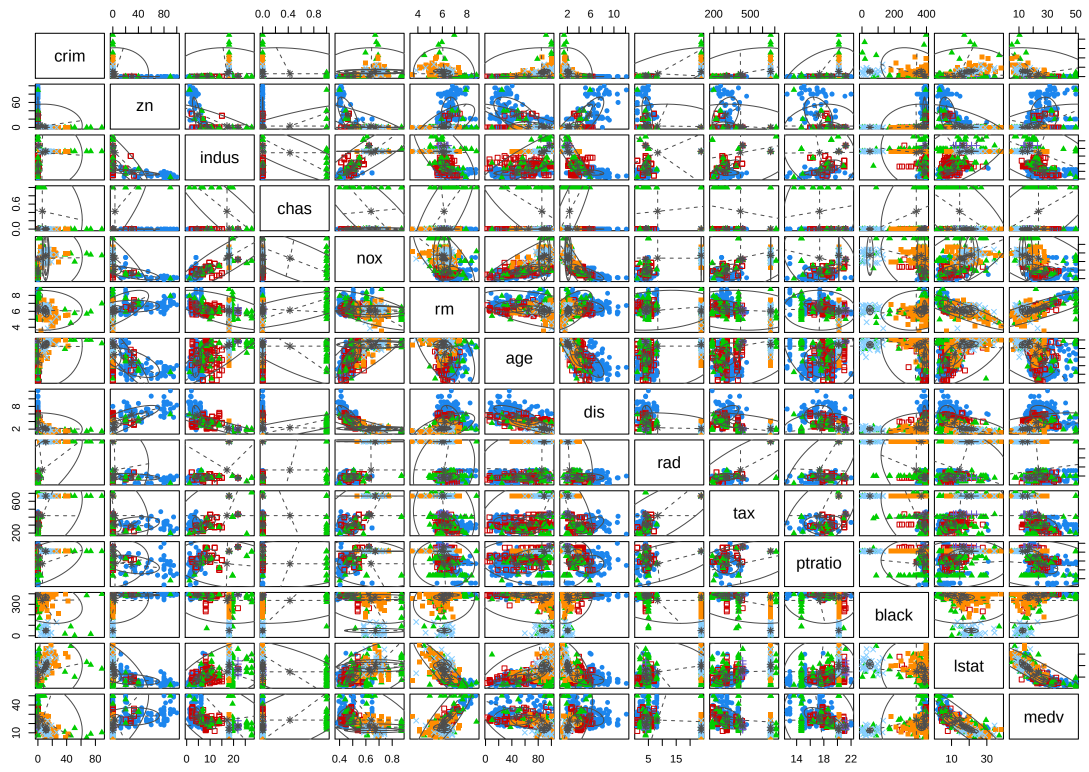


Figure 4.24: **Mclust Boston data- Classification**

To sum up, CTSC obtains goals results operating on this dataset, too. It partitions data into homogeneous subgroups and reaches the goal of identifying groups where observations within the same group are very similar to each other and different from observations in other groups. Moreover, it is demonstrated that high iterations allows obtaining the best results in terms of difference between means of the groups and of distribution of the not overlapped outcome variable.

### 4.5.4 rent99 dataset

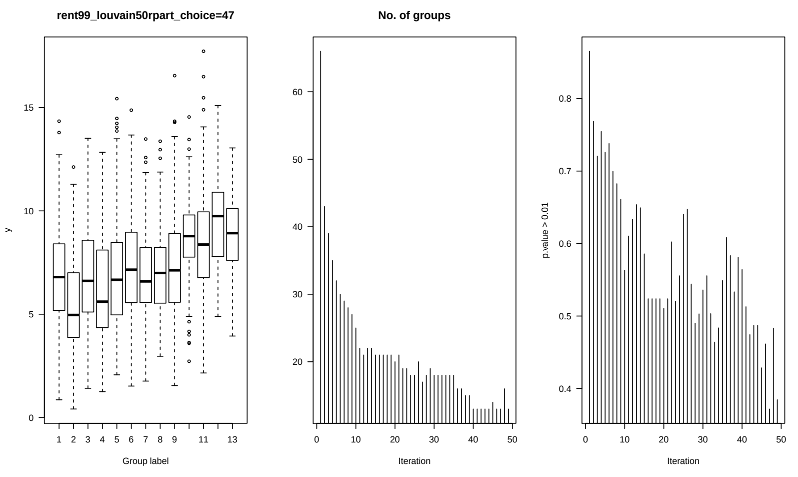
The CTSC is now applied on the rent99 dataset. It contains information on the house rentals in the city of Munich and it has been created in 1999. It is made of 3082 observations and nine variables. The nine variables are:

- *rent*, the monthly net rent per month (in Euro);
- *rentsqm*, the net rent per month per square meter (in Euro);
- *area*, the living area in square meters;

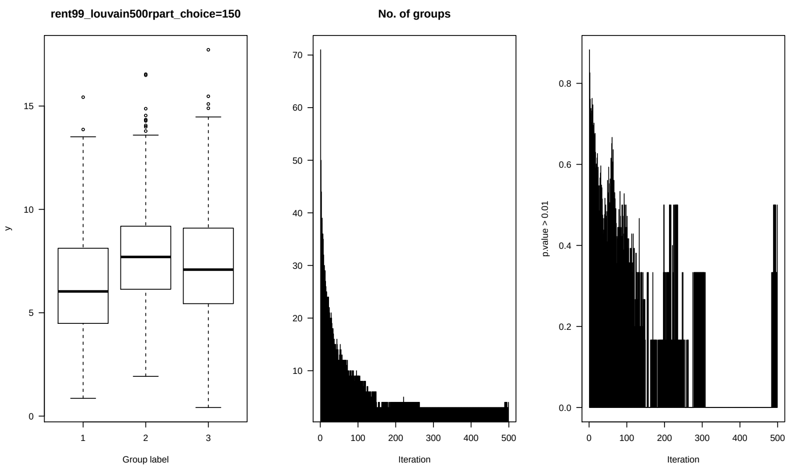
- *yearc*, the year of construction;
- *location*, the quality of the location. It is measured through a factor indicating whether the location is an average location (1), a good location (2) or a top location (3);
- *bath*, the quality of the bathroom(s). it is defined through a factor indicating whether the bath facilities are standard (0) or premium (1);
- *kitchen*, the quality of the kitchen. It is defined through a dichotomous variable equal to 0 for standard quality or 1 for for premium quality;
- *cheating*, the central heating. It is another dichotomous variable, equal to 0 if the house doesn't have a central heating system, and equal to 1 if, on the other side, the house has one;
- *district*, the different districts in Munich.

In a first phase, it has been decided to do not consider the variable *district* in the model, not considered relevant for this analysis, and the variable *rentsqm*, because highly associated with the outcome variable *rent*. As made before for the Boston dataset, CTSC is applied on rent99 using the different values of $B$ and its different combinations of the algorithms.
The number of identified groups decreases when the iterations number decreases (Appendix A- Figures A.7 to A.12). However, this reduction becomes significant only after the 100th iteration. At the same time, the value of *p-value* generally decreases, but not as significant as for the Boston dataset. In fact, this reduction is not steady and the *p-value* increases and decreases continuously in time.
The *Louvain* community detection algorithm is firstly applied, combined with the CART and the GBM (Figures Appendix A- from A.7 to A.8, Figures 4.25 and 4.26). In the first case, the left plot in Figure A.7 shows how the groups number decreases in the best partition and becomes equal to 3 after the 200*th* iteration. Moreover, the distributions of the outcome variables for the different groups and the different partitions are similar only for high values of $B$. In fact, the obtained box plots can be overlapped only after $B > 400$. This result seems to suggest that it is better to use high values of $B$, greater than 500, for big datasets. The same results are obtained with the GBM in Figure 4.26: the groups number decreases for high values of $B$, specifically from 10 to 4. Moreover, the distributions of the outcome variables for each groups are similar in some cases (left box in Figure 4.26), as for instance for the best partitions in $B = 100$ and $B = 200$ and for the best partitions in $B = 300$ and $B = 400$.
Additionally, it is interesting to underline how the combination with CART and *Louvain* for

high levels of iterations gives the best results in terms of a partition with high percentage of low *p-values* (right box in Appendix A - Figure A.7) .

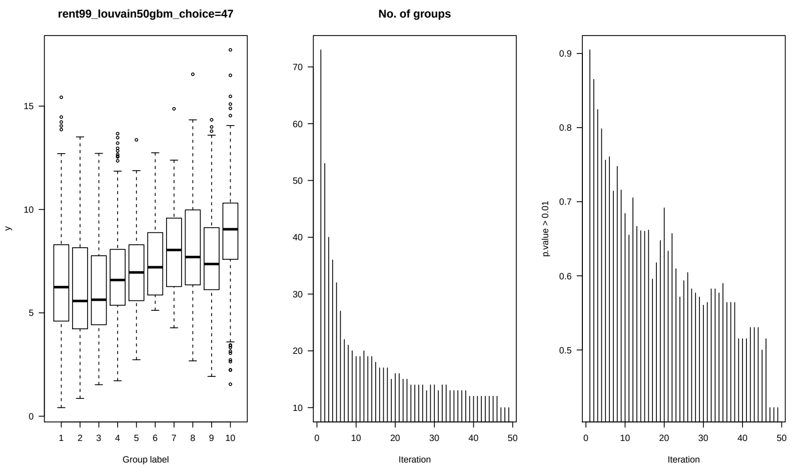


(a) *50 iteration*

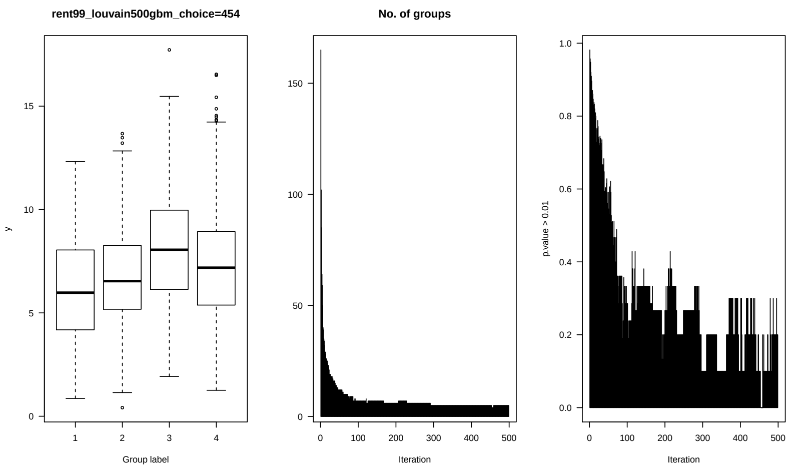


(b) *500 iteration*

Figure 4.25: **Algorithm applied in rent99 including CART and Louvain - Best partitions**



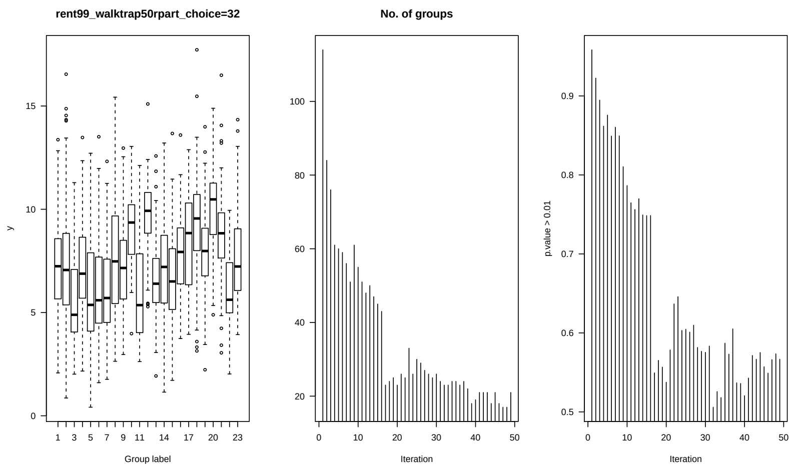
(a) *50 iteration*



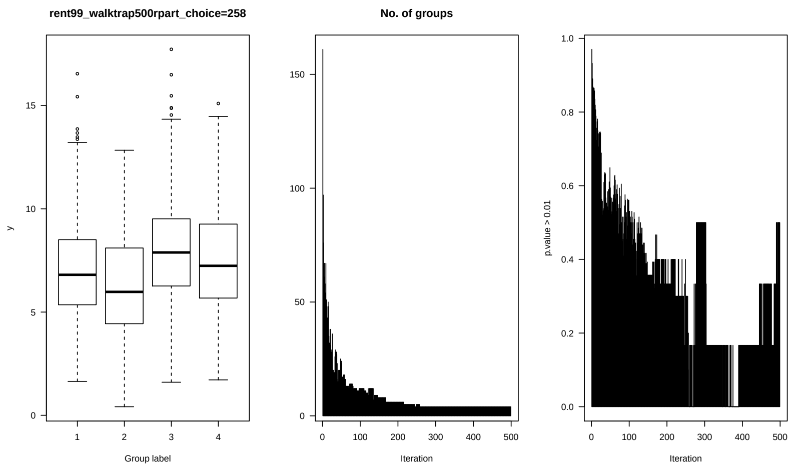
(b) *500 iteration*

Figure 4.26: **Algorithm applied on rent99 including GBM and Louvain - Best partitions**

Later, *Walktrap* is applied as a community detection algorithm. It is firstly combined with CART. Figure 4.27 shows how the identified groups number decreases until 4 after the 300th iteration (central box); the distributions of the outcome variable are roughly equal in the best iterations (left box); and the percentage of partitions with a *p-value* greater the 0.01 firstly decreases and, after, increases (right box).
Secondly, the *Walktrap* is combined with the GBM. The analysis of Figure 4.28 highlights how the number of groups is generally bigger, compared to the results obtained applying CART. However, this quantity is steadier when GBM is applied.

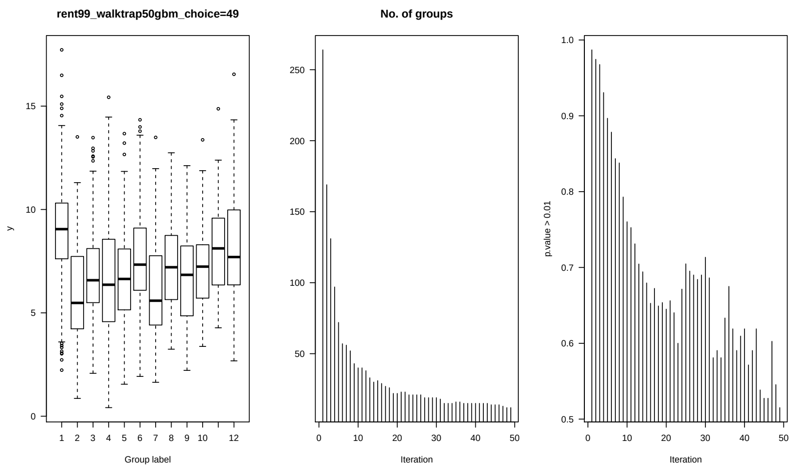


(a) *50 iteration*

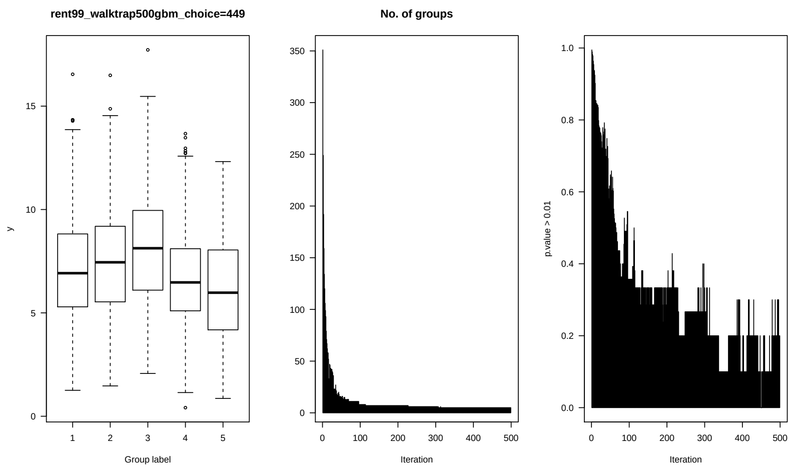


(b) *500 iteration*

Figure 4.27: **Algorithm applied on rent99 including CART and Walktrap - Best partitions**



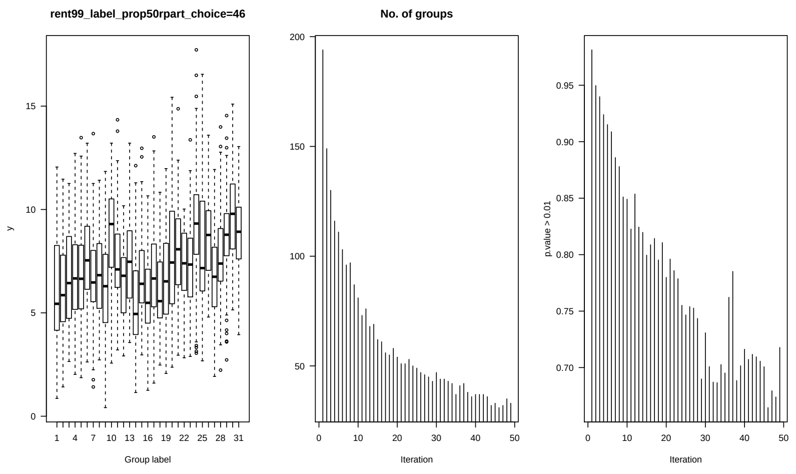
(a) *50 iteration*



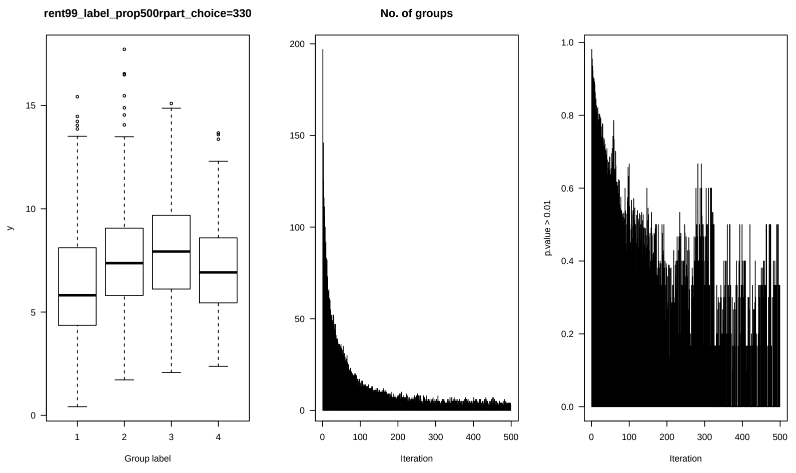
(b) *500 iteration*

Figure 4.28: **Algorithm applied on rent99 including GBM and Walktrap - Best partitions**

The *Label_prop* community detection algorithm is now applied and combined with CART and GBM (Figures 4.29 and 4.30). The results highlight a high instability, taking into account both the quantity of identified groups and the partition with a lowest *p-value* (Appendix A - Figure A.11 to A.12).
Figure A.11 shows the results of the combination with CART: the quantity of groups is equal to 4 after the $200th$ iteration (central box), and the distributions of the outcome variable are partially overlapped (left box). The combination with GBM results in a higher groups number in the best partition, the amount of groups equal to four is obtained only in $B = 500$, as shown in Figure A.12. These results show a less instability in the GBM case.

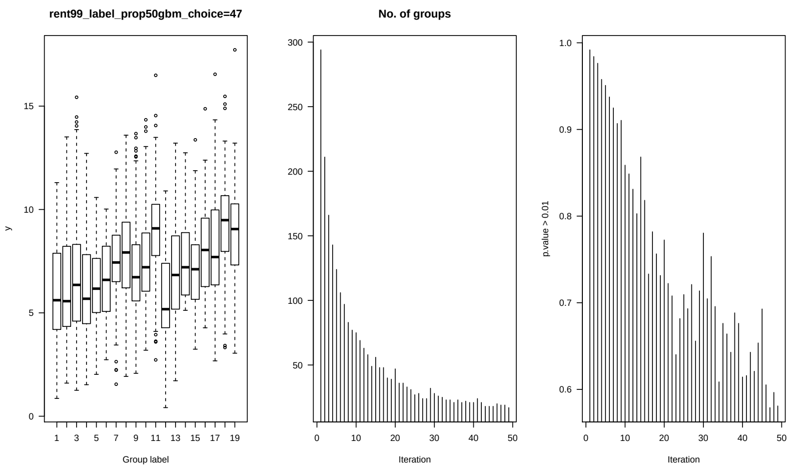


(a) *50 iteration*

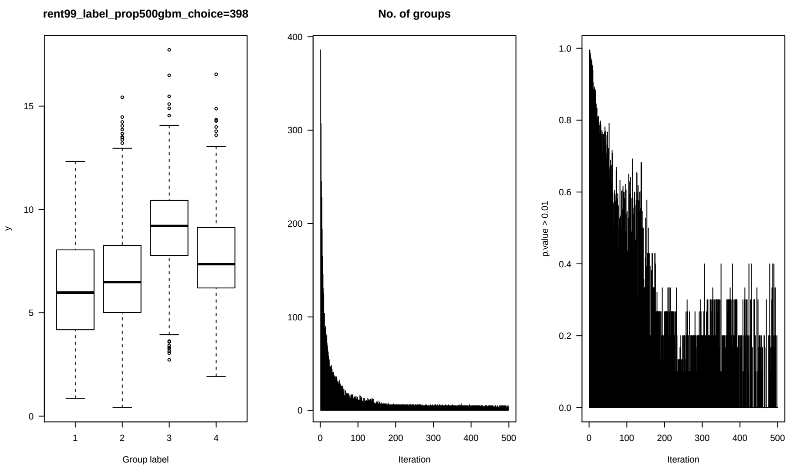


(b) *500 iteration*

Figure 4.29: **Algorithm applied on rent99 including CART and Label Propagation - Best partitions**



(a) *50 iteration*



(b) *500 iteration*

Figure 4.30: **Algorithm applied on rent99 including GBM and Label Propagation - Best partitions**

Then, the percentage of partitions with a lowest value of *p-value* for each combination of CTSC has been calculated, in oder to evaluate the capacity of the algorithm to identify significant partitions. It has been studied how many times the partitions have *p-values* lower than $p = 0.05, 0.01, 0.001, 0.0001, 0.00001$. A value of 1 indicates that all partitions defined for a specific value of $B$ have a lower *p-value*, 100% of the partitions. On the contrary, a value of 0 indicates that all partitions have a *p-value* higher than $p = 0.05, 0.01, 0.001, 0.0001, 0.00001$. Table 4.10 shows that high percentages are obtained only for high values of $B$. In fact, a value bigger than 0.80 is obtained only after the $300th$ iteration.

Generally, the percentage of estimated applying CTSC shows better results compared to those obtained applying the traditional clustering methods. The *Ward.D*, the *Ward.D2* and

the *median* identify only a unique group inside the dataset and for this reason they can not be considered as significative. The *single*, the *complete*, the *average* and the *mcquitty* have a value of 0.67 for $p < 0.05$, and a lower value for the other values of *p-value.* Finally, the percentage for *centroid kmeans* is roughly near to zero.
These results suggest that CTSC is able to work with rather larges dataset, obtaining in some cases better results than the traditional clustering methods.

| Method | Groups | $p_{.05}$ | $p_{.01}$ | $p_{.001}$ | $p_{.0001}$ | $p_{.00001}$ | Average |
|---|---|---|---|---|---|---|---|
| walktrap50rpart | 23 | 0.53 | 0.49 | 0.41 | 0.37 | 0.33 | 0.43 |
| louvain50rpart | 13 | 0.65 | 0.63 | 0.56 | 0.54 | 0.49 | 0.57 |
| label prop50rpart | 31 | 0.38 | 0.34 | 0.29 | 0.25 | 0.20 | 0.29 |
| walktrap100rpart | 10 | 0.64 | 0.60 | 0.53 | 0.49 | 0.36 | 0.52 |
| louvain100rpart | 8 | 0.68 | 0.68 | 0.57 | 0.50 | 0.39 | 0.56 |
| label prop100rpart | 15 | 0.62 | 0.57 | 0.49 | 0.44 | 0.42 | 0.51 |
| walktrap200rpart | 8 | 0.64 | 0.64 | 0.61 | 0.54 | 0.46 | 0.58 |
| louvain200rpart | 6 | 0.80 | 0.80 | 0.67 | 0.60 | 0.53 | 0.68 |
| label prop200rpart | 9 | 0.69 | 0.61 | 0.50 | 0.47 | 0.39 | 0.53 |
| walktrap300rpart | 4 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain300rpart | 4 | **1.00** | **1.00** | 0.67 | 0.50 | 0.50 | 0.73 |
| label prop300rpart | 4 | 0.83 | 0.83 | 0.83 | 0.67 | 0.50 | 0.73 |
| walktrap400rpart | 4 | **1.00** | **1.00** | 0.83 | 0.83 | 0.83 | 0.90 |
| louvain400rpart | 3 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| label prop400rpart | 4 | **1.00** | **1.00** | 0.83 | 0.67 | 0.67 | 0.83 |
| walktrap500rpart | 4 | **1.00** | **1.00** | 0.67 | 0.67 | 0.67 | 0.80 |
| louvain500rpart | 3 | **1.00** | **1.00** | 1.00 | 0.67 | 0.67 | 0.87 |
| label prop500rpart | 4 | **1.00** | **1.00** | 0.83 | 0.67 | 0.67 | 0.83 |
| walktrap50gbm | 12 | 0.55 | 0.48 | 0.42 | 0.33 | 0.27 | 0.41 |
| louvain50gbm | 10 | 0.62 | 0.58 | 0.47 | 0.38 | 0.36 | 0.48 |
| label prop50gbm | 19 | 0.50 | 0.42 | 0.37 | 0.31 | 0.27 | 0.37 |
| walktrap100gbm | 8 | 0.79 | 0.75 | 0.61 | 0.54 | 0.50 | 0.64 |
| louvain100gbm | 6 | 0.87 | 0.87 | 0.80 | 0.60 | 0.60 | 0.75 |
| label prop100gbm | 9 | 0.75 | 0.69 | 0.67 | 0.64 | 0.61 | 0.67 |
| walktrap200gbm | 8 | 0.75 | 0.71 | 0.64 | 0.57 | 0.57 | 0.65 |
| louvain200gbm | 6 | 0.80 | 0.80 | 0.67 | 0.67 | 0.60 | 0.71 |
| label prop200gbm | 6 | 0.80 | 0.80 | 0.73 | 0.67 | 0.60 | 0.72 |
| walktrap300gbm | 6 | 0.87 | 0.87 | 0.67 | 0.67 | 0.60 | 0.73 |
| louvain300gbm | 5 | 0.90 | 0.90 | 0.80 | 0.80 | 0.70 | 0.82 |
| label prop300gbm | 5 | **1.00** | **1.00** | 0.80 | 0.70 | 0.70 | 0.84 |
| walktrap400gbm | 5 | 0.90 | 0.90 | 0.80 | 0.80 | 0.70 | 0.82 |
| louvain400gbm | 5 | 0.90 | 0.90 | 0.80 | 0.80 | 0.60 | 0.80 |
| label prop400gbm | 5 | **1.00** | **1.00** | 0.80 | 0.70 | 0.70 | 0.84 |
| walktrap500gbm | 5 | **1.00** | **1.00** | 0.70 | 0.60 | 0.60 | 0.78 |
| louvain500gbm | 4 | **1.00** | **1.00** | **1.00** | 0.83 | 0.67 | 0.90 |
| label prop500gbm | 4 | **1.00** | **1.00** | **1.00** | 0.83 | 0.83 | 0.93 |
| ward.D | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.80 |
| ward.D2 | 1 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.60 |
| single | 3 | 0.67 | 0.67 | 0.33 | 0.00 | 0.00 | 0.33 |
| complete | 3 | 0.67 | 0.33 | 0.00 | 0.00 | 0.00 | 0.20 |
| average | 3 | 0.67 | 0.33 | 0.33 | 0.33 | 0.00 | 0.33 |
| mcquitty | 3 | 0.67 | 0.00 | 0.00 | 0.00 | 0.00 | 0.13 |
| median | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.80 |
| centroid | 14 | 0.20 | 0.13 | 0.13 | 0.09 | 0.05 | 0.12 |
| kmeans | 3 | 0.33 | 0.00 | 0.00 | 0.00 | 0.00 | 0.07 |

Table 4.10: Application of different algorithm on Rent99 dataset

The analysis of these results shows how the CTSC has been able to identify specific partitions inside the rent99 data. The increase of the value of $B$ supports the reduction of the groups number and of the percentage of partitions with lower value of *p-value*. CTSC obtains better results than the traditional clusters, and it is therefore, in most of the cases, a better clustering method than the traditional ones.
Also in this case, it is applied the R packege Mclust in order to identify the best number of clusters on rent99 data. The results evidence the presence of the two clusters. The number is lower than the results obtained by CTSC, as shown in Figure 4.31.

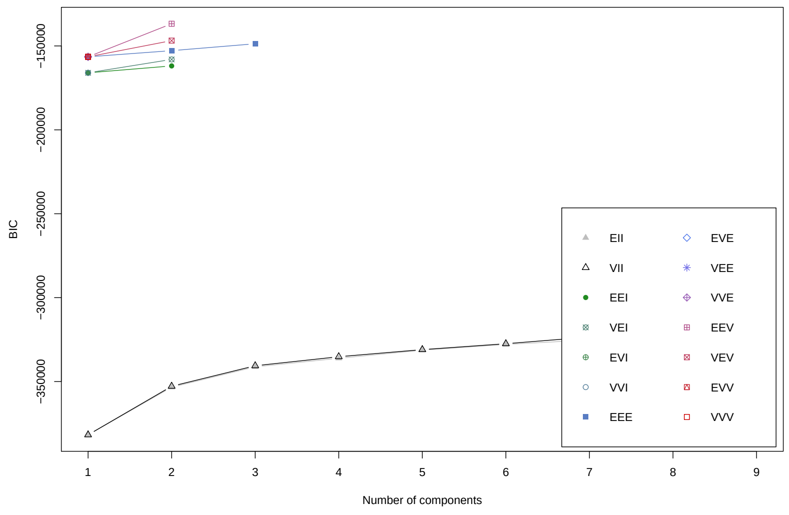


Figure 4.31: **Mclust on rent99- BIC**

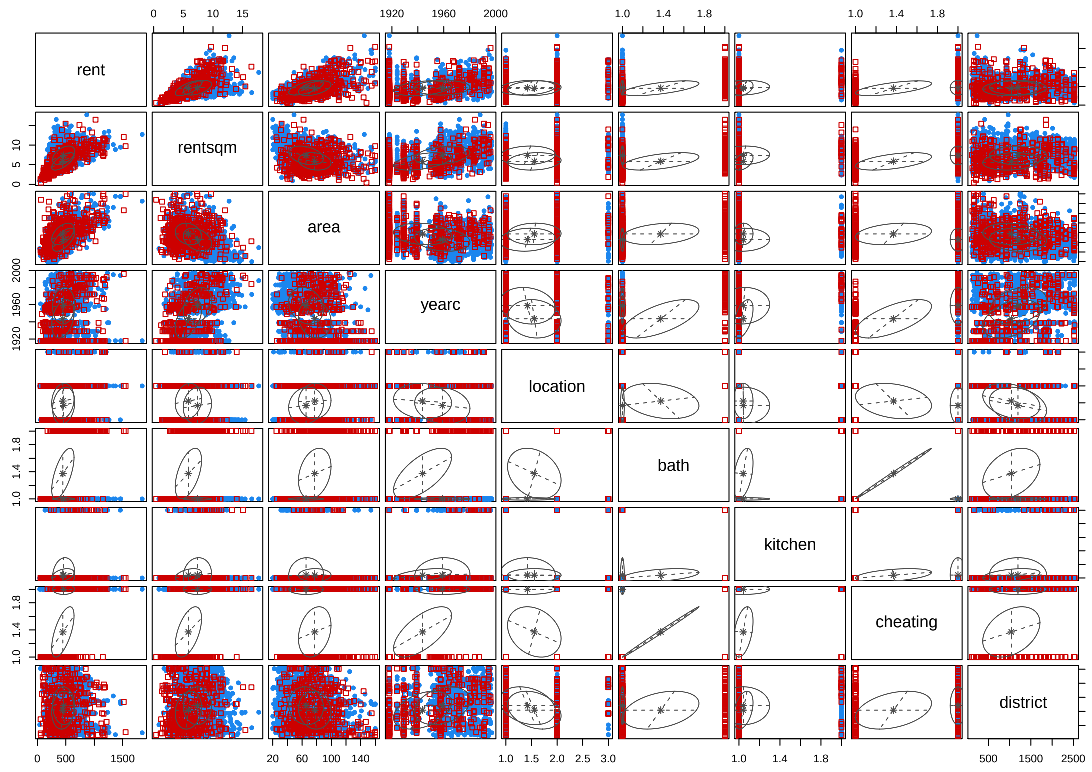


Figure 4.32: **Mclust on rent99- Classification**

Moreover, the results obtained through CTSC on rent99 are compared with the results obtain in two different studies realized respectively by Gertheiss and Tutz (2010) and Tutz and Berger (2018).

The first researchers have proposed a shrinkage methods for categorical predictors. Specifically, Gertheiss and Tutz (2010) have proposed two $L_1$-penalty based methods to select the factor and to cluster categories. In the same way, Tutz and Berger (2018) have focus their attention on the effect of categorical predictors. However, they have proposed a method based on the tree method to find the cluster of factor levels.

In both studies, the researchers have taken into account categorical predictors as for instance the *year of construction, the quality of residential area* and *the district*. They have identified for each predictors a specific number of clusters and defined the different models including the clustered categorical predictors.

The same clustered variables are included in the application of CTSC. Specifically, as done by Gertheiss and Tutz (2010) and Tutz and Berger (2018), the variables *the district* are casted in 25 districts taking into account the rent; the variables *year of construction* is classified in seven groups respect to the relative decade; finally, the *the quality of residential area* presents three specific levels that express the three level: fair, good and excellent.

The CTSC is applied taking into account the combination of CART with the community detection algoritms *Louvain*, *Walktrap* and *Label_prop* The results of the application of CTSC on modified rent99 data evidences a reduction of the number of clusters respect the results obtained with original variables (Figure 4.25, 4.27 and 4.29). Specifically, the number of clusters is equal to two for all community detection algorithms. However, the reduction determines an overlapping of the distributions of outcome variable. The capacity to identify clusters similar w.r.t. an outcome variable seems to be reduce using clustered variables. To sum up, it is possible to state that the use of the variables as defined by Gertheiss and Tutz (2010) and Tutz and Berger (2018) allows to reduce the number of cluster. However, in the same times, it determines a reduction of differences among the identified groups.

# Chapter 5

# Conclusions

The aim of this thesis was to define a new semi-supervised cluster algorithm.
To reach this aim, different aspects have been analyzed. Firstly, the researches on semi-supervised clustering has been studied in order to identify the more suitable framework for this proposal: this framework has been identified in the approach defined by Bair (2013) as *cluster associated with an outcome variable*. Additionally, the different algorithms introduced by other researchers have been analyzed to identify their characteristics, phases, variables and goals. This first phase has been fundamental to comprehend which elements had to be included in the algorithm and how to define its structure.
Later, the attention has been focused on the study of two different methodologies: the tree-based method and the community detection algorithm. The two methodologies have been analyzed in order to identify statistical elements to support the theorization of the algorithm. Firstly, the tree-based method has been studied to find useful algorithms to classify the observations in the same terminal nodes taking into account a specific outcome variable. Two algorithms were chosen: CART and GBM. CART is useful because it is able to identify internally homogenous groups through the recursive partitioning of the feature space and, at the same time, to explain the relationships between the outcome variable and the covariates. GBM has been chosen because it operates step by step to build the estimated outcome, and also because it uses weighted predictors. These two aspects have been considered extremely important for the implementation of the first phase of the algorithm.
Secondly, the attention has posed on the study of community detection in networks in order to evaluate the ability of this algorithm to work as a clustering algorithm. The statement of Arruda et al. (2012), de Oliveira et al. (2008), Granell et al (2011, 2012) have influenced the specification of the algorithm and the choice of the community detection algorithm instead of the traditional clustering methods.
Three specific community detection algorithms have been chosen for their characteristic: *Louvain*, *Walktrap* and *Label_prop*.

The different methodologies have been joined together to shape the proposal of this thesis: an iterative approach of semi-supervised clustering called *Community Detection Tree-Based Algorithm for Semi-supervised Clustering.*

Three phases compose the CTSC. The first one is the pre-training phase. In this phase, the tree classifier is applied as many times as the number of the covariates. Later, the trees generated after each iteration are used both to attribute the initial weights to the observations and to the variables, and to define a proximity matrix. This matrix plays a fundamental role in the CTSC: it indicates how many times one generic observation $i$ is classified with another generic observation $j$, and also gives information about the proximity of the observations providing a measure of the strength of the their relationship. The second phase is the training phase. As happens in first phase, the trees are iteratively built and the proximity matrix is defined. The proximity matrix becomes the input of the third phase: clustering, where the community detection algorithms are implemented.

In order to evaluate the CTSC ability to reach the fixed goals, CTSC has been applied on simulated data and three different datasets. The simulated data are defined taking into account different elements as qualitative and quantitative outcome variables, different number and kinds of covariates, different level of perturbation and overlapping and the presence of noise variable. The results of the application of CTSC show how the semi supervised algorithm works better when the outcome variable is quantitative, the variables are not overlapped and the noise variable are not included in the model. Moreover, the different level of perturbation can influence the capacity of CTCS to identify clusters in case the for quantitative outcome variable.

Later, it is considered three real datasets. The first is a small dataset, manually collected in 2016: the Unesco websites dataset. The goal was to understand the capacity of the CTSC to define correctly the proximity matrix and to identify clusters actually associated with the quantiative outcome variable *Age of websites* and the qualitative outcome variable *Countries.* The results of the application proved the ability of the CTSC to work as a cluster algorithm and to identify clusters associated with the outcome variable.

The second dataset is the Boston data. In this case, CTSC has been applied introducing some new elements: both CART and GBM have been applied. The results of the application underline how the algorithm is able to partition the data in groups that are similar internally w.r.t. the outcome variables.

Later, CTSC has been applied on a third dataset, the rent99. In this case, it has been more difficult for the CTSC to obtain stable results. The best results in terms of number of groups and lowest *pvalue* are obtained only increasing the number of iterations.

The results obtained through the CTSC are compared with the results obtained by application of traditional cluster algorithms. The comparison evidences a dissimilarity in the results and, above all, a greater capacity of CTSC to define cluster w.r.t. a specific outcome variable, reaching the prefixed aim. To sum up, the CTSC demonstrated capable of identifying the clusters, in general, and the cluster with respect to an outcome variable, in particular.
Moreover, it is clear how the proximity matrix,and its transformation in adjacency matrix, is a correct input for the community detection algorithm. The ability to measure the proximity among observations makes the matrix a useful base for the application of modularity, the estimation of the $r$ distance and the propagation of the labels inside the dataset. The advantages of the application of CTSC are that:

- it can be implemented using quantitative and qualitative outcome variables, as well as quantitative and qualitative covariates. This aspect makes the algorithm an important tool for the analysis of different datasets and for the application on various research areas;
- it uses all covariates included in the dataset, not only a subset of them, consequently giving the possibility to consider the whole dataset;
- it combines different trees and community detection algorithms in an adaptive manners.

The CTSC is an innovative semi-supervised clustering method also because it uses a community detection algorithm instead of the traditional cluster methods. In fact, the cluster problem is transformed into a community detection problem.
Some drawbacks are identified as for instance the reduced capacity of CTSC to define groups when the level of perturbation and the overlapping among observations is high and the necessity to improve the number of iterations for big datasets. These elements evidence the necessity to improve the algorithm. Moreover, the analysis of the results suggests that some improvements of CTSC are possible. Firstly, new classifiers, as for instance the random forest, could be used to improve the definition of homogenous groups. At the same time, other community detection algorithms could be considered to provide new combinations, more adaptable to different phenomena and datasets.
The CTSC could (and should) be adjusted in order to be adapted and applied to specific topics. For instance, the CTSC could be applied on medical data, like the dataset analyzed by Bair and Tibshirani (2004), to underline similarities and differences in the results. Also,

the number of iterations in the analyses of big dataset could be improved, to better comprehend if the change of this parameter could improve the analysis.

# Appendix A

# Tables and graphs

## A.1 Simulated data

Table A.1: Simulated data: qualitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 2 | 4 | 1.0 | 100 | 0.95 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |

Table A.1: Simulated data: qualitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.77 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.81 |
| walktrap | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.77 |
| louvain | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.77 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.54 |
| walktrap | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.86 |
| louvain | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.96 |
| label_prop | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.88 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.84 |
| louvain | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.86 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.84 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.92 |
| louvain | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.64 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.81 |
| walktrap | 0.05 | 2 | 6 | 0 | 1.0 | 100 | 0.92 |
| louvain | 0.05 | 2 | 6 | 0 | 1.0 | 100 | 0.92 |
| label_prop | 0.05 | 2 | 6 | 0 | 1.0 | 100 | 0.77 |
| walktrap | 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.84 |
| label_prop | 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.84 |
| walktrap | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.83 |

Table A.1: Simulated data: qualitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.83 |
| walktrap | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.75 |
| louvain | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| label_prop | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.75 |
| walktrap | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.90 |
| louvain | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.84 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.92 |
| walktrap | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.83 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.88 |
| walktrap | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.77 |
| louvain | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.77 |
| label_prop | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.70 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.83 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.83 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.83 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.92 |

Table A.2: Simulated data: qualitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 2 | 4 | 1.0 | 100 | 0.95 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 0.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 0.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |

Table A.2: Simulated data: qualitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 4 | 0 | 1.0 | 100 | 0.45 |
| label_prop | 0,0001 | 4 | 4 | 0 | 1.0 | 100 | 0.45 |
| walktrap | 0,0001 | 4 | 4 | 0 | 0.5 | 100 | 0.29 |
| louvain | 0,0001 | 4 | 4 | 0 | 0.5 | 100 | 0.29 |
| label_prop | 0,0001 | 4 | 4 | 0 | 0.5 | 100 | 0.29 |
| walktrap | 0,0001 | 4 | 4 | 0 | 0.0 | 100 | 0.29 |
| louvain | 0,0001 | 4 | 4 | 0 | 0.0 | 100 | 0.29 |
| label_prop | 0,0001 | 4 | 4 | 0 | 0.0 | 100 | 0.29 |
| walktrap | 0,0001 | 4 | 4 | 2 | 1.0 | 100 | 0.45 |
| louvain | 0,0001 | 4 | 4 | 2 | 1.0 | 100 | 0.48 |
| label_prop | 0,0001 | 4 | 4 | 2 | 1.0 | 100 | 0.45 |
| walktrap | 0,0001 | 4 | 4 | 2 | 0.5 | 100 | 0.29 |
| louvain | 0,0001 | 4 | 4 | 2 | 0.5 | 100 | 0.41 |
| label_prop | 0,0001 | 4 | 4 | 2 | 0.5 | 100 | 0.29 |
| walktrap | 0,0001 | 4 | 4 | 2 | 0.0 | 100 | 0.29 |

Table A.2: Simulated data: qualitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0,0001 | 4 | 4 | 2 | 0.0 | 100 | 0.29 |
| label_prop | 0,0001 | 4 | 4 | 2 | 0.0 | 100 | 0.2 |
| walktrap | 0,0001 | 4 | 4 | 4 | 1.0 | 100 | 0.45 |
| louvain | 0,0001 | 4 | 4 | 4 | 1.0 | 100 | 0.52 |
| label_prop | 0,0001 | 4 | 4 | 4 | 1.0 | 100 | 0.45 |
| walktrap | 0,0001 | 4 | 4 | 4 | 0.5 | 100 | 0.29 |
| louvain | 0,0001 | 4 | 4 | 4 | 0.5 | 100 | 0.44 |
| label_prop | 0,0001 | 4 | 4 | 4 | 0.5 | 100 | 0.29 |
| walktrap | 0,0001 | 4 | 4 | 4 | 0.0 | 100 | 0.29 |
| louvain | 0,0001 | 4 | 4 | 4 | 0.0 | 100 | 0.29 |
| label_prop | 0,0001 | 4 | 4 | 4 | 0.0 | 100 | 0.29 |
| walktrap | 0,0001 | 4 | 6 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 2 | 0.5 | 100 | 0.45 |
| louvain | 0,0001 | 4 | 6 | 2 | 0.5 | 100 | 0.67 |
| label_prop | 0,0001 | 4 | 6 | 2 | 0.5 | 100 | 0.45 |
| walktrap | 0,0001 | 4 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0,0001 | 4 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0,0001 | 4 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0,0001 | 4 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0.38 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0.38 |
| walktrap | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0.63 |
| louvain | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0.63 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0.63 |
| walktrap | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0.47 |
| louvain | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0.47 |
| label_prop | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0.29 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0.63 |
| louvain | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0.51 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0.51 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0.63 |

Table A.2: Simulated data: qualitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0.63 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0.63 |
| walktrap | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0.47 |
| louvain | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0.47 |
| label_prop | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0.29 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0.52 |
| louvain | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0.42 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0.52 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0.36 |
| louvain | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0.41 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0.47 |
| walktrap | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0.34 |
| louvain | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0.42 |
| label_prop | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0.36 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0.17 |
| louvain | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0.27 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0.17 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0.23 |
| louvain | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0.49 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0.21 |
| walktrap | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0.49 |
| louvain | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0.49 |
| label_prop | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0.46 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0.15 |
| louvain | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0.14 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0.03 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0.14 |
| louvain | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0.25 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0.17 |
| walktrap | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0.48 |
| louvain | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0.50 |
| label_prop | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0.43 |
| walktrap | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0.17 |
| louvain | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0.27 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0.12 |
| walktrap | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0.24 |
| louvain | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0.30 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0.15 |
| walktrap | 0.05 | 4 | 6 | 0 | 1.0 | 100 | 0.42 |
| louvain | 0.05 | 4 | 6 | 0 | 1.0 | 100 | 0.42 |
| label_prop | 0.05 | 4 | 6 | 0 | 1.0 | 100 | 0.42 |
| walktrap | 0.05 | 4 | 6 | 0 | 0.5 | 100 | 0.56 |
| louvain | 0.05 | 4 | 6 | 0 | 0.5 | 100 | 0.56 |
| label_prop | 0.05 | 4 | 6 | 0 | 0.5 | 100 | 0.56 |
| walktrap | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0.33 |
| louvain | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0.44 |
| label_prop | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0.45 |
| walktrap | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0.42 |
| louvain | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0.42 |
| label_prop | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0.42 |

Table A.2: Simulated data: qualitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0.40 |
| louvain | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0.47 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0.34 |
| walktrap | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0.41 |
| louvain | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0.42 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0.28 |
| walktrap | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0.46 |
| louvain | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0.46 |
| label_prop | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0.42 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0.52 |
| louvain | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0.52 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0.39 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0.43 |
| louvain | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0.34 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0.40 |

Table A.3: Simulated data: qualitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.23 |
| walktrap | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.70 |
| louvain | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.70 |
| label_prop | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.70 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.22 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.28 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.07 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 0.22 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 0.37 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 0.11 |
| walktrap | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.70 |
| louvain | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.70 |
| label_prop | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.70 |
| walktrap | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.05 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.38 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.25 |
| walktrap | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 0.27 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 0.48 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 0.27 |
| walktrap | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.70 |
| louvain | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.73 |
| label_prop | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.70 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.25 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.25 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.11 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 0.36 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 0.38 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 0.25 |
| walktrap | 0.0001 | 6 | 6 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 0 | 1.0 | 100 | 1.00 |

Table A.3: Simulated data: qualitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 6 | 6 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 0 | 0.5 | 100 | 0.36 |
| louvain | 0.0001 | 6 | 6 | 0 | 0.5 | 100 | 0.36 |
| label_prop | 0.0001 | 6 | 6 | 0 | 0.5 | 100 | 0.36 |
| walktrap | 0.0001 | 6 | 6 | 0 | 0.0 | 100 | 0.11 |
| louvain | 0.0001 | 6 | 6 | 0 | 0.0 | 100 | 0.20 |
| label_prop | 0.0001 | 6 | 6 | 0 | 0.0 | 100 | 0.11 |
| walktrap | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 0.25 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 0.34 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 0.25 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 0.11 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 0.18 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 0.11 |
| walktrap | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 0.75 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 0.18 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 0.36 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 0.25 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 0.11 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 0.20 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 0.11 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.07 |
| louvain | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.21 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.06 |
| walktrap | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.35 |
| louvain | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.45 |
| label_prop | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.36 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.12 |
| louvain | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.29 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.27 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.26 |
| louvain | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.27 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.03 |
| walktrap | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.31 |
| louvain | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.36 |
| label_prop | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.27 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.11 |
| louvain | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.18 |
| label_prop | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.07 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.12 |
| louvain | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.23 |
| label_prop | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.17 |
| walktrap | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.35 |
| louvain | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.46 |
| label_prop | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.46 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.15 |

Table A.3: Simulated data: qualitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.23 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.14 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.16 |
| louvain | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.32 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.20 |
| walktrap | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.21 |
| label_prop | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.17 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.25 |
| louvain | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.25 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.07 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.28 |
| louvain | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.19 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.15 |
| walktrap | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.10 |
| louvain | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.27 |
| label_prop | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.14 |
| walktrap | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.24 |
| louvain | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.24 |
| label_prop | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.06 |
| walktrap | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.18 |
| louvain | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.26 |
| label_prop | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.09 |
| walktrap | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.16 |
| louvain | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.24 |
| label_prop | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.23 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.22 |
| louvain | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.24 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.21 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.24 |
| louvain | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.25 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.15 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 0.96 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 2 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 0.92 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 0.96 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 0.92 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 0.92 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 0.92 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 2 | 6 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| louvain | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| label_prop | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| walktrap | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| louvain | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| label_prop | 0.05 | 2 | 2 | 0 | 1.0 | 100 | 0.60 |
| walktrap | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.43 |
| louvain | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.38 |
| label_prop | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.43 |
| walktrap | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.55 |
| louvain | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.55 |
| label_prop | 0.05 | 2 | 2 | 0 | 0.5 | 100 | 0.55 |
| walktrap | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.57 |
| louvain | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.42 |
| label_prop | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.57 |
| walktrap | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.57 |
| louvain | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.57 |
| label_prop | 0.05 | 2 | 2 | 0 | 0.0 | 100 | 0.54 |
| walktrap | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.20 |
| louvain | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.20 |
| label_prop | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.20 |
| walktrap | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.48 |
| louvain | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.48 |
| label_prop | 0.05 | 2 | 2 | 2 | 1.0 | 100 | 0.48 |
| walktrap | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.01 |
| louvain | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.57 |
| label_prop | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.42 |
| walktrap | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.05 |
| louvain | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.55 |
| label_prop | 0.05 | 2 | 2 | 2 | 0.5 | 100 | 0.55 |
| walktrap | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.57 |
| louvain | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.46 |
| label_prop | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.35 |
| walktrap | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.51 |
| louvain | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.64 |
| label_prop | 0.05 | 2 | 2 | 2 | 0.0 | 100 | 0.48 |
| label_prop | 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.54 |
| walktrap | 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.54 |
| louvain | 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.54 |
| label_prop | 0.05 | 2 | 4 | 0 | 0.5 | 100 | 0.84 |
| walktrap | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.48 |
| louvain | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.60 |
| label_prop | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.67 |
| walktrap | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.81 |
| louvain | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.88 |
| label_prop | 0.05 | 2 | 4 | 0 | 0.0 | 100 | 0.81 |
| walktrap | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.70 |
| louvain | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.57 |
| label_prop | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.70 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.35 |
| louvain | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.35 |
| label_prop | 0.05 | 2 | 4 | 2 | 1.0 | 100 | 0.88 |
| walktrap | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.46 |
| louvain | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.30 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.38 |
| walktrap | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.74 |
| louvain | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.74 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.5 | 100 | 0.74 |
| walktrap | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.54 |
| louvain | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.48 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.54 |
| walktrap | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.81 |
| louvain | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.81 |
| label_prop | 0.05 | 2 | 4 | 2 | 0.0 | 100 | 0.81 |
| walktrap | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.67 |
| louvain | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.51 |
| label_prop | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.77 |
| walktrap | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.74 |
| louvain | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.38 |
| label_prop | 0.05 | 2 | 4 | 4 | 1.0 | 100 | 0.88 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.54 |
| louvain | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.43 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.64 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.83 |
| louvain | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.60 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.5 | 100 | 0.77 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.48 |
| louvain | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.54 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.54 |
| walktrap | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.81 |
| louvain | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.81 |
| label_prop | 0.05 | 2 | 4 | 4 | 0.0 | 100 | 0.81 |
| louvain | 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.88 |
| label_prop | 0.05 | 2 | 6 | 0 | 0.5 | 100 | 0.92 |
| walktrap | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.70 |
| louvain | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.74 |
| label_prop | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.70 |
| walktrap | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.83 |
| louvain | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.62 |
| label_prop | 0.05 | 2 | 6 | 0 | 0.0 | 100 | 0.88 |
| walktrap | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| louvain | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| label_prop | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.77 |
| walktrap | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.76 |
| louvain | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.76 |
| label_prop | 0.05 | 2 | 6 | 2 | 1.0 | 100 | 0.76 |
| walktrap | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.77 |
| louvain | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.57 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.88 |

Table A.4: Simulated data: quantitative outcome, k=2

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.96 |
| louvain | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.96 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.5 | 100 | 0.96 |
| walktrap | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.46 |
| louvain | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.33 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.48 |
| walktrap | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.88 |
| louvain | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.88 |
| label_prop | 0.05 | 2 | 6 | 2 | 0.0 | 100 | 0.96 |
| walktrap | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.77 |
| louvain | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.81 |
| label_prop | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.77 |
| walktrap | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.75 |
| louvain | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.75 |
| label_prop | 0.05 | 2 | 6 | 4 | 1.0 | 100 | 0.75 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.74 |
| louvain | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.81 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.70 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.84 |
| louvain | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 0.66 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.38 |
| louvain | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.19 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.67 |
| walktrap | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.92 |
| louvain | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.96 |
| label_prop | 0.05 | 2 | 6 | 4 | 0.0 | 100 | 0.96 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.0001 | 4 | 2 | 0 | 1.0 | 100 | 0,73 |
| label_prop | 0.0001 | 4 | 2 | 0 | 1.0 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 0 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 0 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |
| louvain | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |
| label_prop | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |
| label_prop | 0.0001 | 4 | 2 | 2 | 1.0 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 0,73 |
| louvain | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 0,95 |
| label_prop | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 0,73 |
| louvain | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 2 | 0.5 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,73 |
| louvain | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,75 |
| label_prop | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,73 |
| louvain | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,73 |
| label_prop | 0.0001 | 4 | 2 | 4 | 1.0 | 100 | 0,73 |
| walktrap | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 0,78 |
| louvain | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 0,78 |
| label_prop | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 0,71 |
| walktrap | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 2 | 4 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 0 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 0,70 |
| louvain | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 0,90 |
| label_prop | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 0,70 |
| walktrap | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 0 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 0 | 0.0 | 100 | 1,00 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 4 | 4 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 0 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 0 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 0 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 2 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 1.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 0.5 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| walktrap | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| louvain | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| label_prop | 0.0001 | 4 | 6 | 4 | 0.0 | 100 | 1,00 |
| walktrap | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,13 |
| louvain | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,38 |
| label_prop | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,46 |
| walktrap | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,13 |
| louvain | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,38 |
| label_prop | 0.05 | 4 | 2 | 0 | 1.0 | 100 | 0,46 |
| walktrap | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,51 |
| louvain | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,39 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,14 |
| walktrap | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,48 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,39 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.5 | 100 | 0,14 |
| walktrap | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,45 |
| louvain | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,45 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,79 |
| walktrap | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,80 |
| louvain | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 0 | 0.0 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,18 |
| louvain | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,18 |
| label_prop | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,33 |
| walktrap | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,18 |
| louvain | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,33 |
| label_prop | 0.05 | 4 | 2 | 2 | 1.0 | 100 | 0,33 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,68 |
| louvain | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,15 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,77 |
| louvain | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,77 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.5 | 100 | 0,77 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,45 |
| louvain | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,45 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,45 |
| walktrap | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,80 |
| louvain | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 2 | 0.0 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,38 |
| louvain | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,38 |
| label_prop | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,38 |
| walktrap | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,39 |
| louvain | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,39 |
| label_prop | 0.05 | 4 | 2 | 4 | 1.0 | 100 | 0,33 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,15 |
| louvain | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,22 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,20 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,80 |
| louvain | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,40 |
| louvain | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,52 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,26 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |
| louvain | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |
| louvain | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.5 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,40 |
| louvain | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,52 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,26 |
| walktrap | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |
| label_prop | 0.05 | 4 | 2 | 4 | 0.0 | 100 | 0,80 |
| walktrap | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,41 |
| louvain | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,41 |
| label_prop | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,41 |
| walktrap | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,58 |
| louvain | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,58 |
| label_prop | 0.05 | 4 | 4 | 0 | 1.0 | 100 | 0,49 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,33 |
| louvain | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,36 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,41 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,40 |
| louvain | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,41 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.5 | 100 | 0,32 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,29 |
| louvain | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,35 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,04 |
| walktrap | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,26 |
| louvain | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,48 |
| label_prop | 0.05 | 4 | 4 | 0 | 0.0 | 100 | 0,24 |
| walktrap | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,41 |
| louvain | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,52 |
| label_prop | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,41 |
| walktrap | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,58 |
| louvain | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,58 |
| label_prop | 0.05 | 4 | 4 | 2 | 1.0 | 100 | 0,58 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,27 |
| louvain | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,35 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,32 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,34 |
| louvain | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,57 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.5 | 100 | 0,56 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,29 |
| louvain | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,30 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,33 |
| walktrap | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,29 |
| louvain | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,41 |
| label_prop | 0.05 | 4 | 4 | 2 | 0.0 | 100 | 0,23 |
| walktrap | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,41 |
| louvain | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,52 |
| label_prop | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,41 |
| walktrap | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,58 |
| louvain | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,58 |
| label_prop | 0.05 | 4 | 4 | 4 | 1.0 | 100 | 0,58 |
| walktrap | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,31 |
| louvain | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,50 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,33 |
| walktrap | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,45 |
| louvain | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,60 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.5 | 100 | 0,37 |

Table A.5: Simulated data: quantitative outcome, k=4

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,30 |
| louvain | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,30 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,23 |
| walktrap | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,37 |
| louvain | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,38 |
| label_prop | 0.05 | 4 | 4 | 4 | 0.0 | 100 | 0,18 |
| louvain | 0.05 | 4 | 6 | 0 | 0.5 | 100 | 0,37 |
| label_prop | 0.05 | 4 | 6 | 0 | 0.5 | 100 | 0,29 |
| walktrap | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,14 |
| louvain | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,20 |
| label_prop | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,09 |
| walktrap | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,30 |
| louvain | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,13 |
| label_prop | 0.05 | 4 | 6 | 0 | 0.0 | 100 | 0,07 |
| walktrap | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,30 |
| louvain | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,41 |
| label_prop | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,06 |
| walktrap | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,40 |
| louvain | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,32 |
| label_prop | 0.05 | 4 | 6 | 2 | 1.0 | 100 | 0,31 |
| walktrap | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,06 |
| louvain | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,09 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,10 |
| walktrap | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,19 |
| louvain | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,21 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.5 | 100 | 0,07 |
| walktrap | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,24 |
| louvain | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,29 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,08 |
| walktrap | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,18 |
| louvain | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,11 |
| label_prop | 0.05 | 4 | 6 | 2 | 0.0 | 100 | 0,10 |
| walktrap | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,33 |
| louvain | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,36 |
| label_prop | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,25 |
| walktrap | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,11 |
| louvain | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,34 |
| label_prop | 0.05 | 4 | 6 | 4 | 1.0 | 100 | 0,25 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,41 |
| louvain | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,18 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,18 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,19 |
| louvain | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,21 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.5 | 100 | 0,26 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,26 |
| louvain | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,25 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,07 |
| walktrap | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,19 |
| louvain | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,19 |
| label_prop | 0.05 | 4 | 6 | 4 | 0.0 | 100 | 0,12 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.0001 | 6 | 2 | 0 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 0 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 0 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 0 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.26 |
| louvain | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.26 |
| label_prop | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.26 |
| walktrap | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.84 |
| louvain | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.84 |
| label_prop | 0.0001 | 6 | 2 | 0 | 0.5 | 100 | 0.84 |
| walktrap | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.40 |
| louvain | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.33 |
| label_prop | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.32 |
| walktrap | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.64 |
| louvain | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.64 |
| label_prop | 0.0001 | 6 | 2 | 0 | 0.0 | 100 | 0.64 |
| walktrap | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 2 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.28 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.48 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.28 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.64 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.64 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.5 | 100 | 0.64 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.36 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.41 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.44 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| walktrap | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.61 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.38 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.47 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.30 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.59 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.59 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.49 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.36 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.41 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.44 |
| walktrap | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| louvain | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| label_prop | 0.0001 | 6 | 2 | 2 | 0.0 | 100 | 0.60 |
| walktrap | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| walktrap | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| louvain | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.58 |
| label_prop | 0.0001 | 6 | 2 | 4 | 1.0 | 100 | 0.61 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.34 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.5 | 100 | 0.80 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.38 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.47 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.30 |
| walktrap | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.59 |
| louvain | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.59 |
| label_prop | 0.0001 | 6 | 2 | 4 | 0.0 | 100 | 0.49 |
| walktrap | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| louvain | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| label_prop | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| walktrap | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| louvain | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| label_prop | 0.0001 | 6 | 4 | 0 | 1.0 | 100 | 0.41 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.32 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.34 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.32 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.14 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.52 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.5 | 100 | 0.14 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 0.98 |
| walktrap | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |
| louvain | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |
| label_prop | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |
| walktrap | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |
| louvain | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |
| label_prop | 0.0001 | 6 | 4 | 2 | 1.0 | 100 | 0.51 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.17 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.17 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.13 |
| walktrap | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.17 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.17 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.5 | 100 | 0.30 |
| walktrap | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| louvain | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| label_prop | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| walktrap | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| louvain | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| label_prop | 0.0001 | 6 | 4 | 4 | 1.0 | 100 | 0.51 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.98 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.98 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 0.98 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 4 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 0 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 0 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 2 | 0.0 | 100 | 1.00 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| walktrap | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 1.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.5 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| walktrap | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| louvain | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.0001 | 6 | 6 | 4 | 0.0 | 100 | 1.00 |
| label_prop | 0.05 | 6 | 2 | 0 | 0.5 | 100 | 0.28 |
| walktrap | 0.05 | 6 | 2 | 0 | 0.5 | 100 | 0.63 |
| louvain | 0.05 | 6 | 2 | 0 | 0.5 | 100 | 0.63 |
| label_prop | 0.05 | 6 | 2 | 0 | 0.5 | 100 | 0.63 |
| walktrap | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.43 |
| louvain | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.35 |
| label_prop | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.43 |
| walktrap | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.39 |
| louvain | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.23 |
| label_prop | 0.05 | 6 | 2 | 0 | 0.0 | 100 | 0.30 |
| walktrap | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| louvain | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| label_prop | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| walktrap | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| louvain | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| label_prop | 0.05 | 6 | 2 | 2 | 1.0 | 100 | 0.56 |
| walktrap | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.47 |
| louvain | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.48 |
| label_prop | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.48 |
| walktrap | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.43 |
| louvain | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.52 |
| label_prop | 0.05 | 6 | 2 | 2 | 0.5 | 100 | 0.28 |
| walktrap | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.24 |
| louvain | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.35 |
| label_prop | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.37 |
| walktrap | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.41 |
| louvain | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.41 |
| label_prop | 0.05 | 6 | 2 | 2 | 0.0 | 100 | 0.23 |
| walktrap | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.56 |
| louvain | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.56 |
| label_prop | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.56 |
| walktrap | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.59 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| louvain | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.59 |
| label_prop | 0.05 | 6 | 2 | 4 | 1.0 | 100 | 0.59 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.12 |
| louvain | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.19 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.00 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.12 |
| louvain | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.12 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.5 | 100 | 0.55 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.33 |
| louvain | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.38 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.33 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.39 |
| louvain | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.23 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.36 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.33 |
| walktrap | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.39 |
| louvain | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.23 |
| label_prop | 0.05 | 6 | 2 | 4 | 0.0 | 100 | 0.36 |
| walktrap | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.22 |
| louvain | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.35 |
| label_prop | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.22 |
| walktrap | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.22 |
| louvain | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.35 |
| label_prop | 0.05 | 6 | 4 | 0 | 1.0 | 100 | 0.22 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.24 |
| louvain | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.24 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.02 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.08 |
| louvain | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.17 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.5 | 100 | 0.09 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.49 |
| louvain | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.49 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.49 |
| walktrap | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.03 |
| louvain | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.54 |
| label_prop | 0.05 | 6 | 4 | 0 | 0.0 | 100 | 0.54 |
| walktrap | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.24 |
| louvain | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.34 |
| label_prop | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.24 |
| walktrap | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.34 |
| louvain | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.36 |
| label_prop | 0.05 | 6 | 4 | 2 | 1.0 | 100 | 0.22 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.21 |
| louvain | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.21 |
| label_prop | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.11 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.21 |
| louvain | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.20 |
| label_prop | 0.05 | 6 | 4 | 2 | 0.5 | 100 | 0.10 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.30 |
| louvain | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.30 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.30 |
| walktrap | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.30 |
| label_prop | 0.05 | 6 | 4 | 2 | 0.0 | 100 | 0.26 |
| walktrap | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.24 |
| louvain | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.38 |
| label_prop | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.38 |
| walktrap | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.34 |
| louvain | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.36 |
| label_prop | 0.05 | 6 | 4 | 4 | 1.0 | 100 | 0.22 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.21 |
| louvain | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.21 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.13 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.15 |
| louvain | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.21 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.5 | 100 | 0.10 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.29 |
| louvain | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.29 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.20 |
| walktrap | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.13 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.11 |
| louvain | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.13 |
| label_prop | 0.05 | 6 | 4 | 4 | 0.0 | 100 | 0.11 |
| walktrap | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.09 |
| label_prop | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.26 |
| walktrap | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.09 |
| label_prop | 0.05 | 6 | 6 | 0 | 1.0 | 100 | 0.26 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.17 |
| louvain | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.18 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.09 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.23 |
| louvain | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.22 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.5 | 100 | 0.15 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.08 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.04 |
| walktrap | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.20 |
| louvain | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.20 |
| label_prop | 0.05 | 6 | 6 | 0 | 0.0 | 100 | 0.06 |
| walktrap | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.07 |
| louvain | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.29 |
| label_prop | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.01 |
| walktrap | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.16 |
| louvain | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.11 |
| label_prop | 0.05 | 6 | 6 | 2 | 1.0 | 100 | 0.22 |
| walktrap | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.09 |
| louvain | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.10 |

Table A.6: Simulated data: quantitative outcome, k=6

| algorithm | overlap | group | Nvar | noise | Varqua | iteration | Rand Index |
|---|---|---|---|---|---|---|---|
| label_prop | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.04 |
| walktrap | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.18 |
| louvain | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.13 |
| label_prop | 0.05 | 6 | 6 | 2 | 0.5 | 100 | 0.25 |
| walktrap | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.07 |
| louvain | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.15 |
| label_prop | 0.05 | 6 | 6 | 2 | 0.0 | 100 | 0.09 |
| walktrap | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.08 |
| louvain | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.29 |
| label_prop | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.01 |
| walktrap | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.16 |
| louvain | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.16 |
| label_prop | 0.05 | 6 | 6 | 4 | 1.0 | 100 | 0.00 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.06 |
| louvain | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.03 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.11 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.16 |
| louvain | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.13 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.5 | 100 | 0.13 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.11 |
| louvain | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.14 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.06 |
| walktrap | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.12 |
| louvain | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.16 |
| label_prop | 0.05 | 6 | 6 | 4 | 0.0 | 100 | 0.03 |

## A.2 Unesco website data

| Variable | Characteristic |
|---|---|
| Contact and support | Search engine |
| | Site map |
| | Foreign languages |
| | Phone number of information offices |
| | Email address |
| | Links to social networks |
| | Generic application form |
| | F.A.Q. |
| | App for them mobile devices (phone and tablet) |
| Adequacy of reported information (Brand) | Specific brand - site of UNESCO |
| | UNESCO's brand |
| | Brand of countries |
| | Additional brand(s) |

Table A.7: Observed characteristic of websites and variable obtained from them

| Variable | Characteristic |
|---|---|
| Adequacy of reported information (Tourism) | Download depliant |
|  | Page flipper |
|  | Gallery images |
|  | Video |
|  | Web TV |
|  | The virtual tour |
|  | Tour guides contacts |
|  | Webcam live |
|  | What to do (activities to be performed) |
|  | Events calendar |
|  | Events calendar with booking facilities |
|  | Information for customers with specific needs |
|  | Information for owners of animal |
|  | Information on how to get |
|  | Information on how to move in the destination - bus |
|  | Information on how to move in the destination - car rental |
|  | Information on how to move in the destination - taxi service |
|  | Map(s) |
|  | Receptivity - List of accommodation facilities |
|  | Receptivity - availability of accommodation |
|  | Online booking |
|  | Restaurants |
|  | Shopping - food and wine products |
|  | Shopping - typical handmade products |
|  | Shopping - commercial centres |
|  | Offers/Last minute promotions |
|  | Advice on how designing a trip |
|  | Information on local travel agencies |

Table A.8: Observed characteristic of websites and variable obtained from them

| Variable | Characteristic |
|---|---|
| Adequacy of reported information(News) | Press releases |
| | Press review |
| | News |
| Relational Skills | Space/wall to share experiences and photos of tourists |
| | Blog |
| | TripAdvisor |
| | Newsletter |
| Other Information - Internal Communication | Data on tourism |
| | Information about the UNESCO sites |
| | Information on regional policies |
| | News and press releases for tourism operators |
| | Operator training area |
| Adequacy of reported information (External Information) | Links to local tourism organizations |
| | Links to municipality sites |
| | Links to sites of the local organization |
| | Links to the sites of other territorial authorities |

Table A.9: Observed characteristic of websites and variable obtained from them

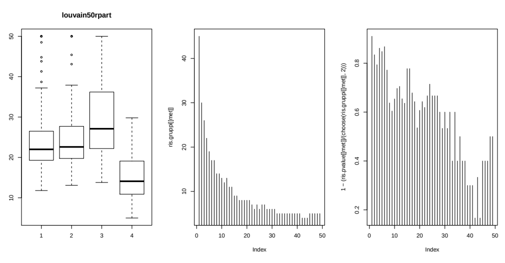
| Indicator | Definition |
|---|---|
| Global Rank | It measures how a website is doing relative to all other sites on the web over the past 3 months. |
| Rank In | It measures how a website ranks in a particular country relative to other websites over the past month. |
| Bounce Rate | The percentage of visitors that leave the website after visiting only one page. |
| Daily Pageviews per Visitor | The average number of pages that are seen by visitor per day. |
| Daily Time on Site | A measure of the time spent by a visitor on the website. |
| Search traffic (also called *organic search engine traffic*) | The percentage of visitors who arrive at a website using search engines. |
| Upstream sites | The indication of which website people visit before visiting a specific website. |

Table A.10: Alexa Indicators

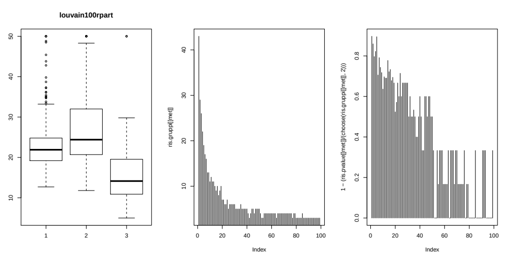
| Area | Focus on | Number of binary variables |
|---|---|---|
| Contact and support | Presence of contacts elements (as phone number and email address) and of links to both apps that can support tourists and to social networks. | 9 |
| Adequacy of Reported Information - Brand | The brand of UNESCO or other brands used inside a website. | 4 |
| Adequacy of Reported Information - Tourism | Tourism information (as accommodation, availability of accommodation, restaurant, touristic activities and services). | 28 |
| Adequacy of Reported Information - External Information | Links to local tourist institutions, tourism product clubs and other public institutions. | 4 |
| Adequacy of Reported Information - News | News and articles informing visitors about events or other activities that take place in the cultural and natural sites. | 3 |
| Relational Skills | Chat, pages where to share photos and videos and where to write reviews. All elements are useful in creating relationships with online visitors. | 4 |
| Other Information/ Internal Communication | Information about the number of visitors, the UNESCO organization and the heritage list, ideas for tourism development, etc. | 5 |

Table A.11: Counting Variables obtained from the presence/absence of specific characteristics of observed websites

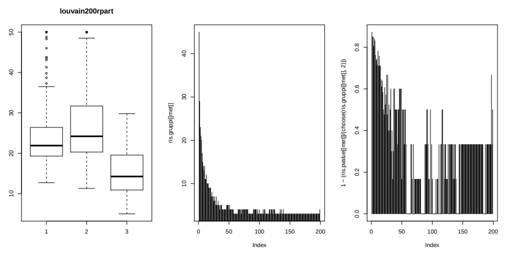
## A.3 Boston data



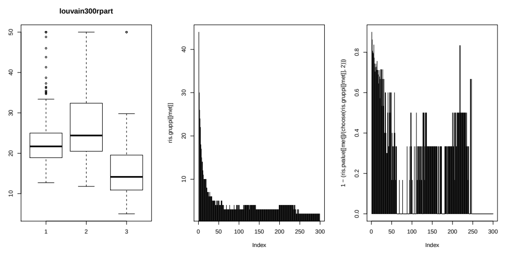
(a) *50th iteration*



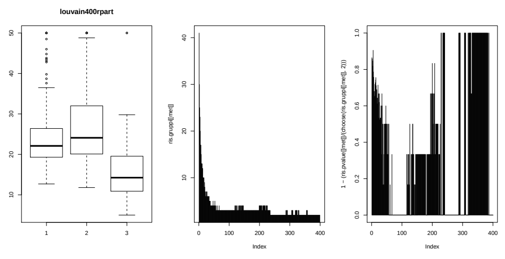
(b) *100th iteration*



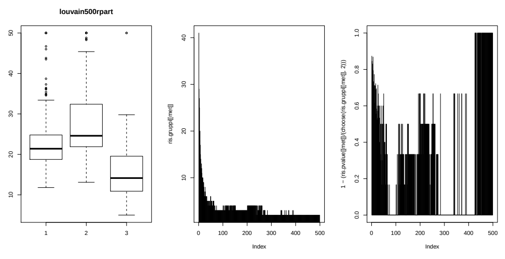
(c) *200th iteration*



(d) *300th iteration*

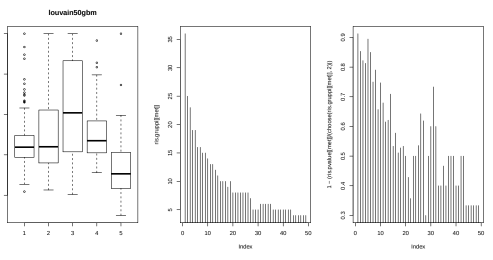


(e) *400th iteration*

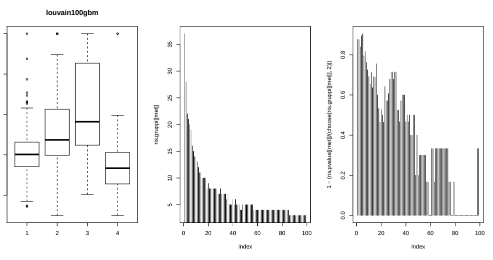


(f) *500th iteration*

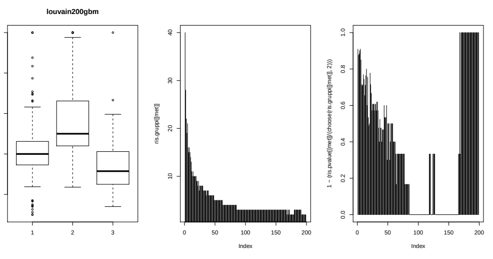
Figure A.1: **CTSC applied on Boston, CART and Louvain**



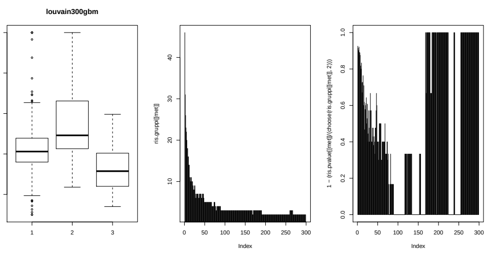
(a) *50th iteration*



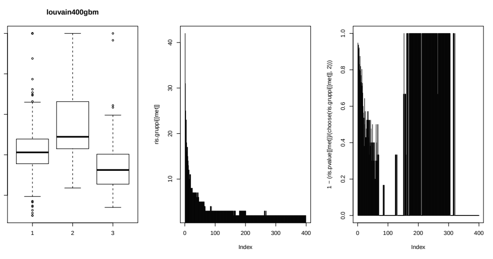
(b) *100th iteration*



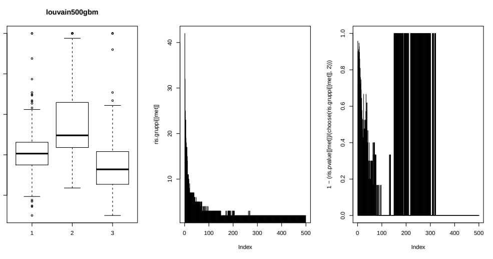
(c) *200th iteration*



(d) *300th iteration*

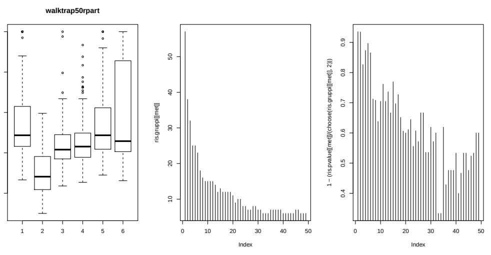


(e) *400th iteration*

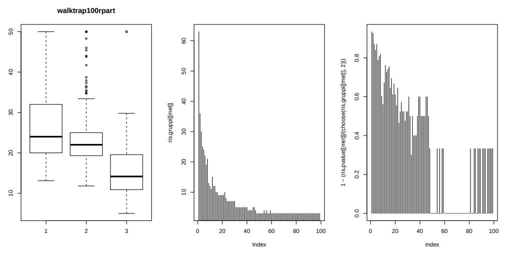


(f) *500th iteration*

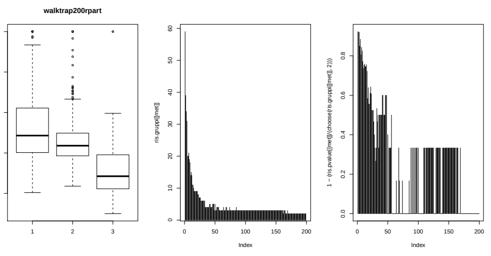
Figure A.2: **CTSC applied on Boston, BGM and Louvain**



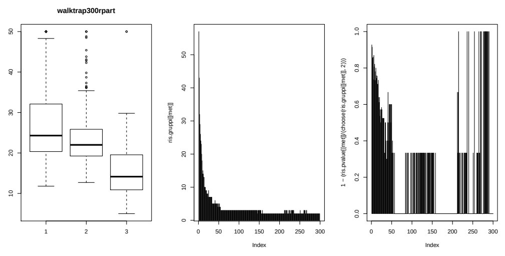
(a) *50th iteration*



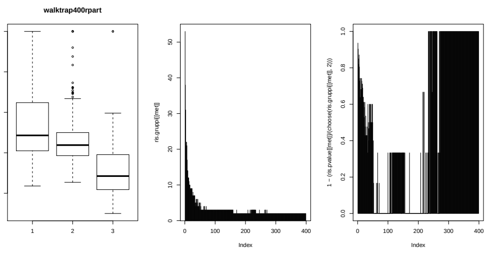
(b) *100th iteration*



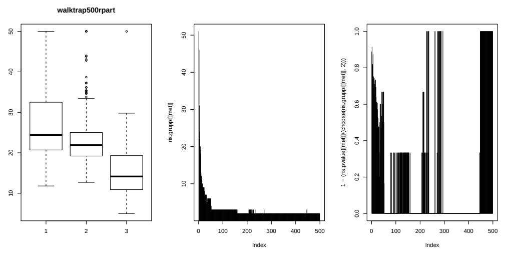
(c) *200th iteration*



(d) *300th iteration*

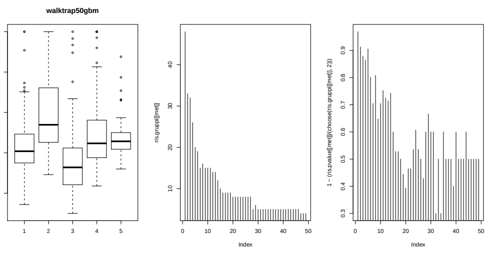


(e) *400th iteration*

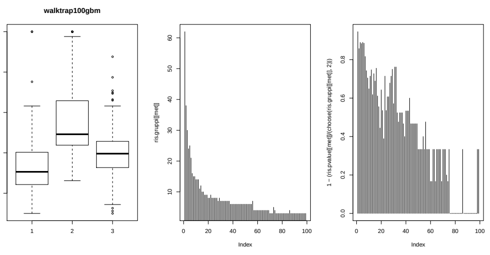


(f) *500th iteration*

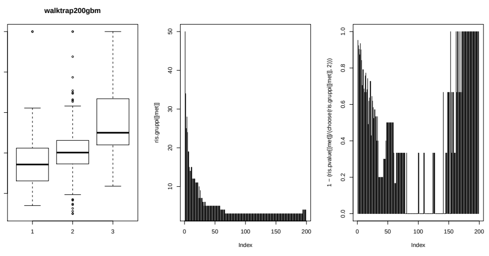
Figure A.3: **CTSC applied on Boston, CART and Walktrap**



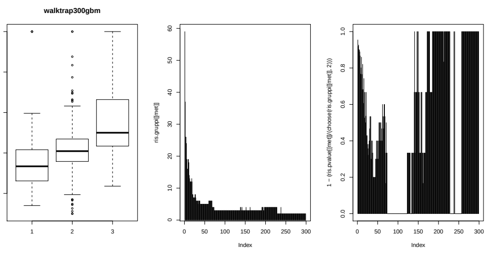
(a) *50th iteration*



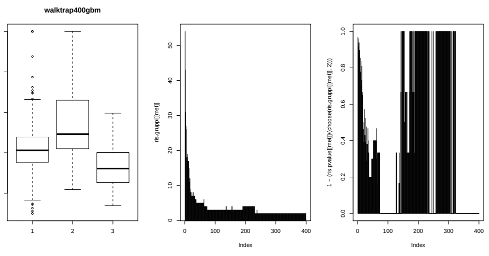
(b) *100th iteration*



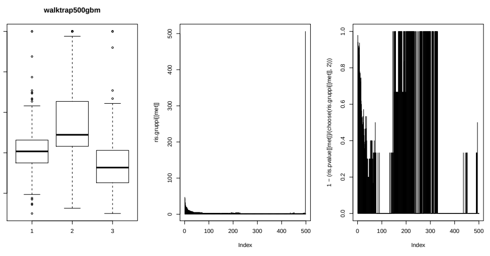
(c) *200th iteration*



(d) *300th iteration*

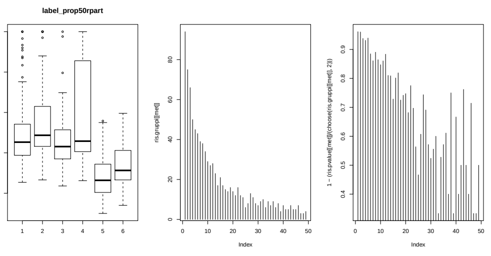


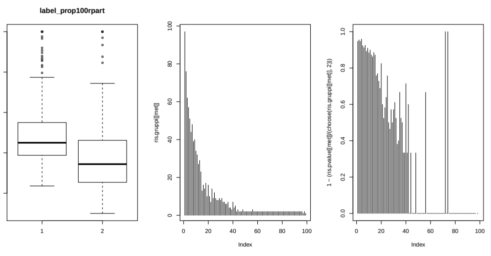
(e) *400th iteration*

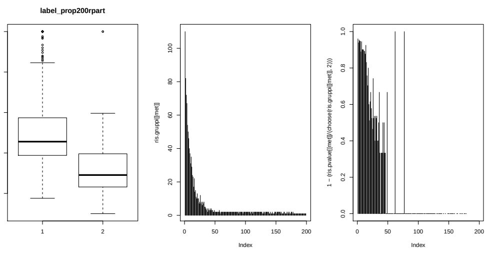


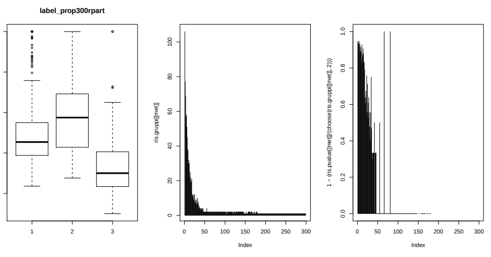
(f) *500th iteration*

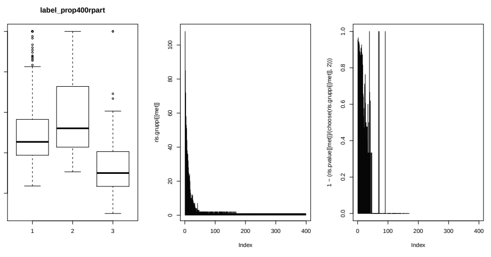
Figure A.4: **CTSC applied on Boston, GBM and Walktrap**


(a) *50th iteration*


(b) *100th iteration*


(c) *200th iteration*


(d) *300th iteration*


(e) *400th iteration*

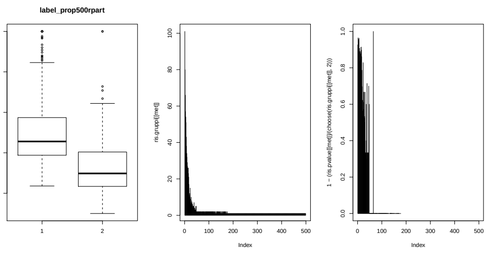
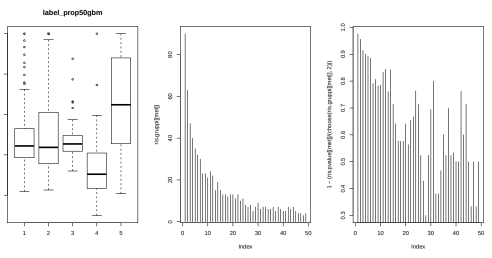
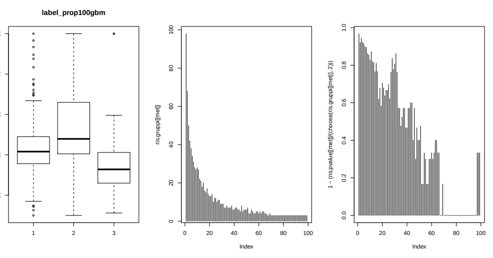

(f) *500th iteration*

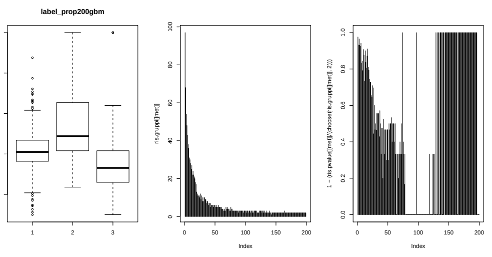
Figure A.5: **CTSC applied on Boston, CART and Label Propagation**



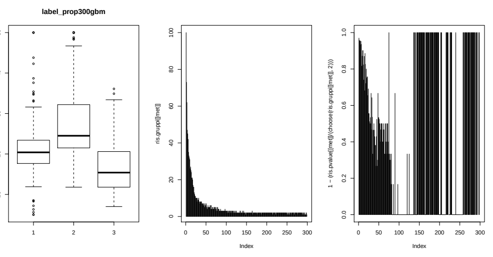
(a) *50th iteration*



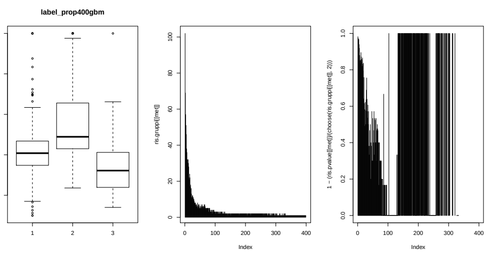
(b) *100th iteration*



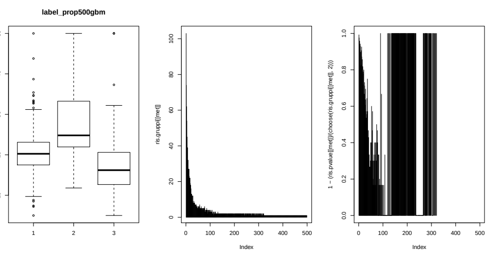
(c) *200th iteration*



(d) *300th iteration*



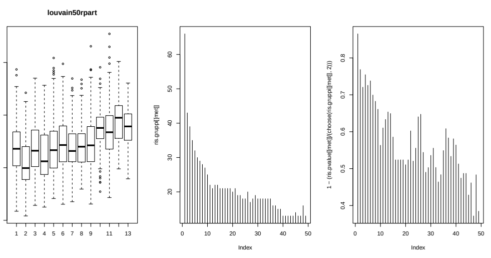
(e) *400th iteration*



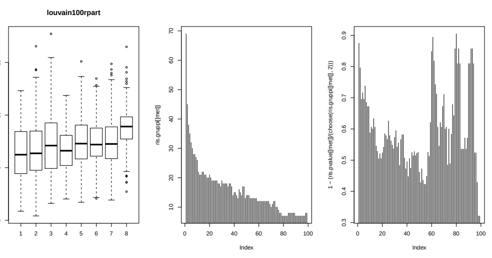
(f) *500th iteration*

Figure A.6: **CTSC applied on Boston, GBM and Label Propagation**

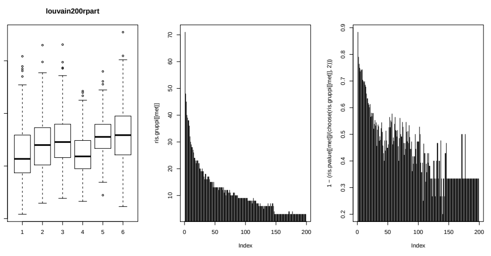
## A.4 rent99



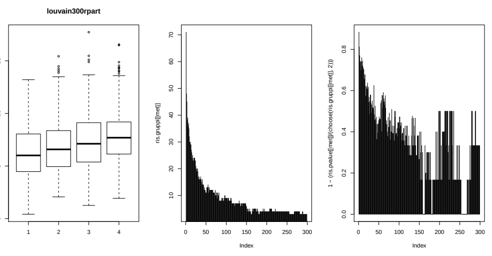
(a) *50th iteration*



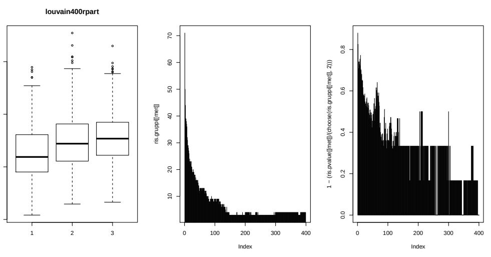
(b) *100th iteration*



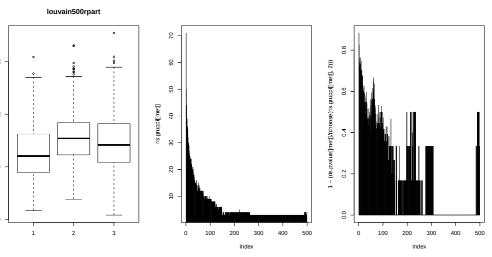
(c) *200th iteration*



(d) *300th iteration*

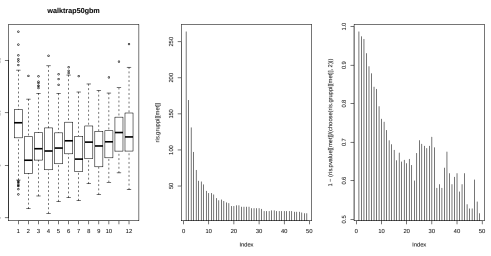


(e) *400th iteration*

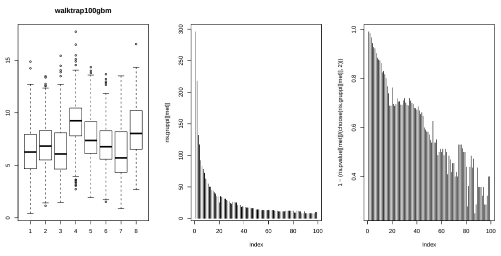


(f) *500th iteration*

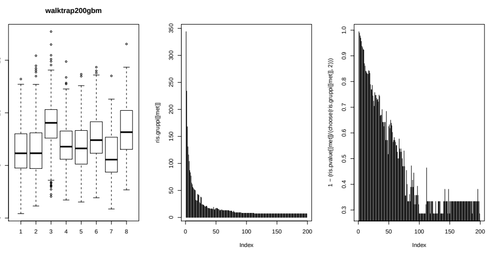
Figure A.7: **CTSC applied on rent99 including CART and Louvain**



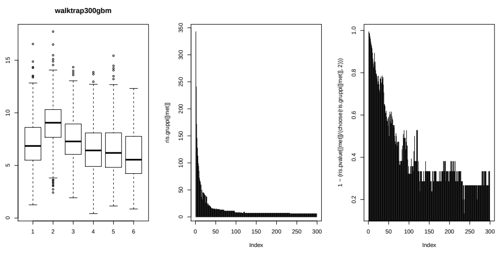
(a) *50th iteration*



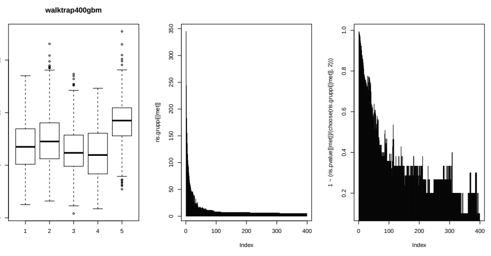
(b) *100th iteration*



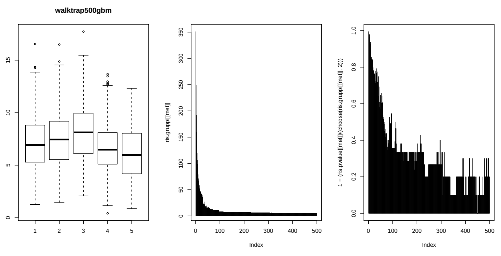
(c) *200th iteration*



(d) *300th iteration*

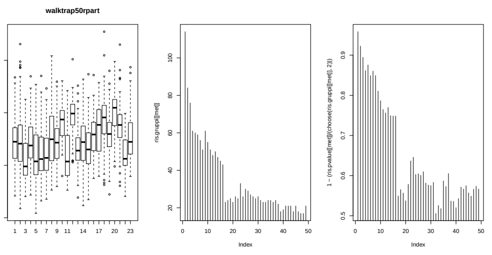


(e) *400th iteration*

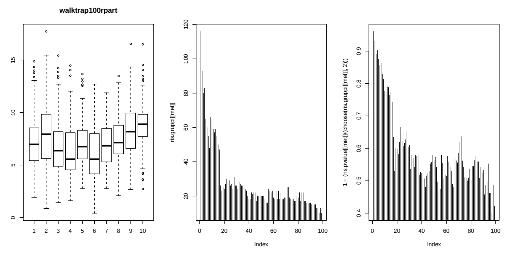


(f) *500th iteration*

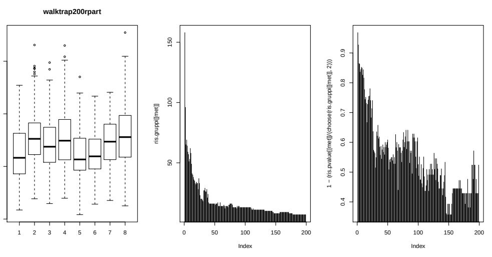
Figure A.8: **CTSC applied on rent99 including GBM and Louvain**



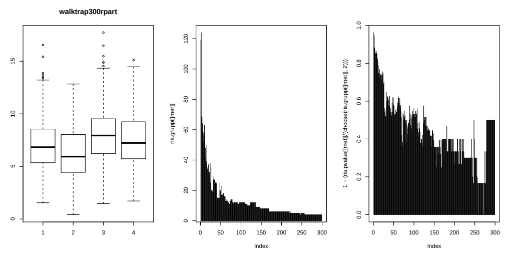
(a) *50th iteration*



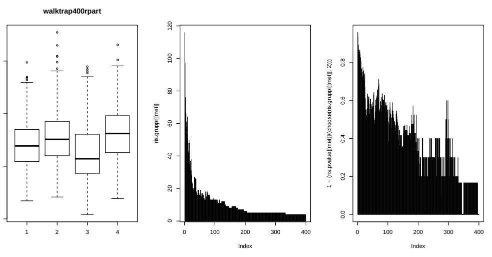
(b) *100th iteration*



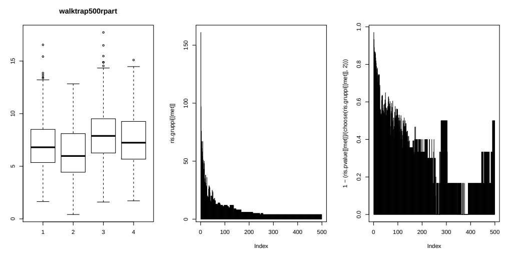
(c) *200th iteration*



(d) *300th iteration*

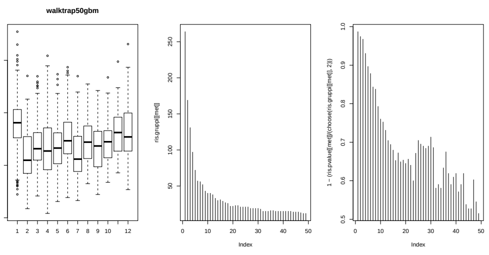


(e) *400th iteration*

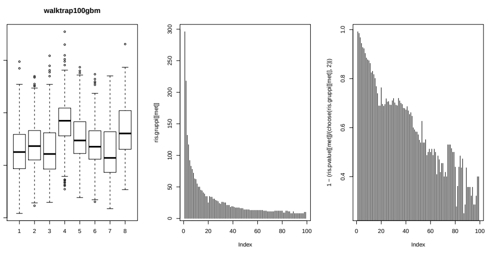


(f) *500th iteration*

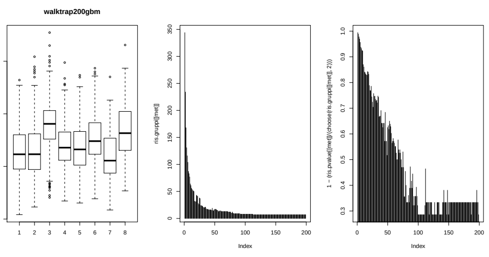
Figure A.9: **CTSC applied on rent99 including CART and Walktrap**



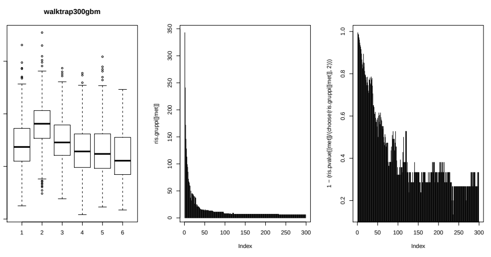
(a) *50th iteration*



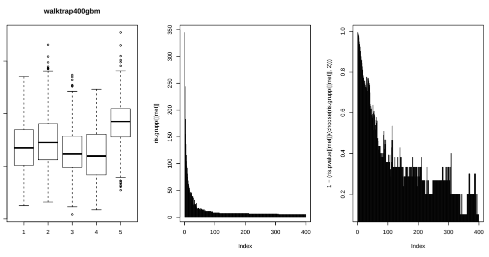
(b) *100th iteration*



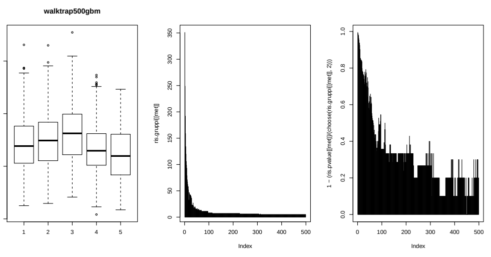
(c) *200th iteration*



(d) *300th iteration*

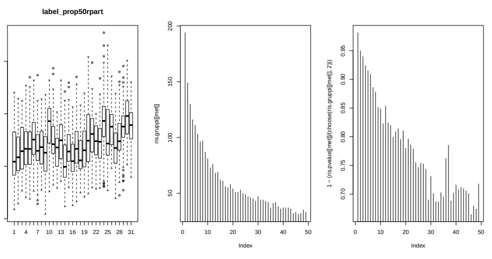


(e) *400th iteration*

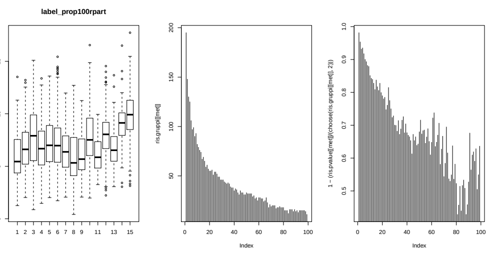


(f) *500th iteration*

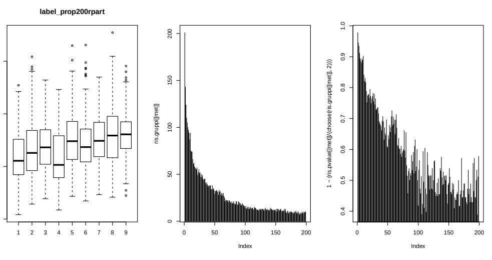
Figure A.10: **CTSC applied on rent99 including GBM and Walktrap**



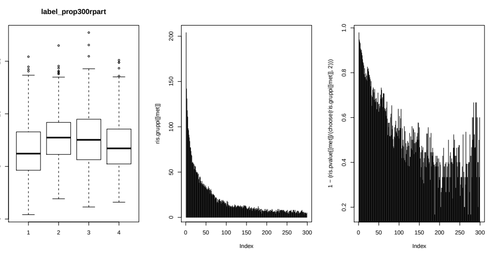
(a) *50th iteration*



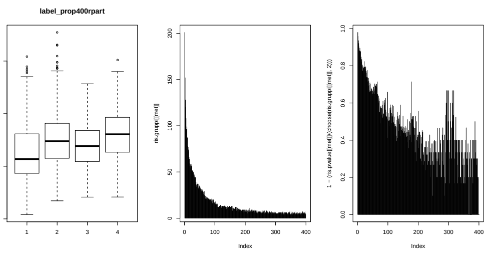
(b) *100th iteration*



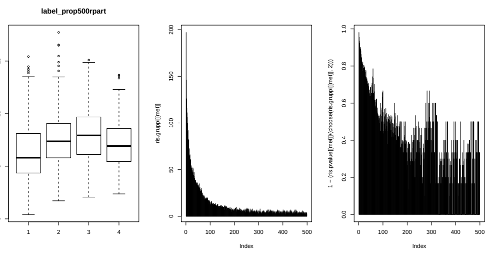
(c) *200th iteration*



(d) *300th iteration*

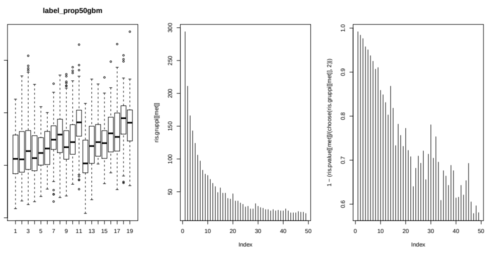


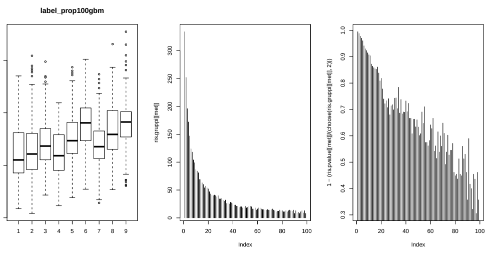
(e) *400th iteration*

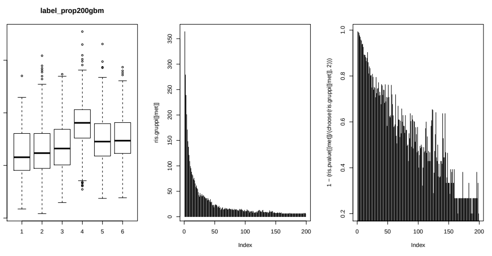


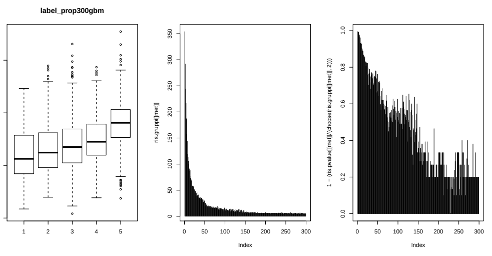
(f) *500th iteration*

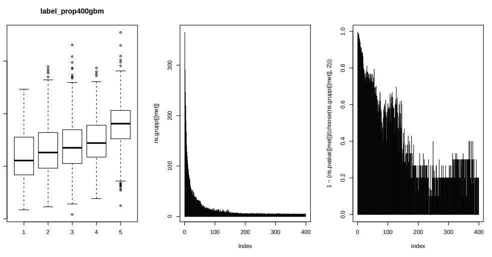
Figure A.11: **CTSC applied on rent99 including CART and Label Propagation**


(a) *50th iteration*


(b) *100th iteration*


(c) *200th iteration*


(d) *300th iteration*


(e) *400th iteration*

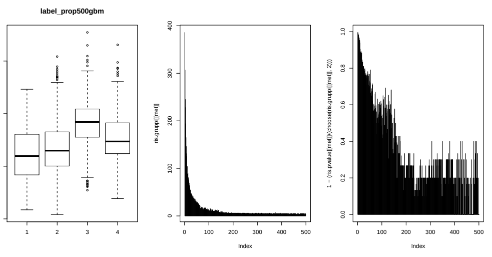

(f) *500th iteration*

Figure A.12: **CTSC applied on rent99 including GBM and Label Propagation**

# Bibliography

[1] Hervé Abdi. The method of least squares. *Encyclopedia of Measurement and Statistics. CA, USA: Thousand Oaks*, 2007.

[2] Steven Abney. *Semisupervised learning for computational linguistics.* CRC Press, 2007.

[3] Gaurav Agarwal and David Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418, 2008.

[4] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: theory, algorithms, and applications.* Prentice hall, 1993.

[5] Richard D Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3(1):113–126, 1973.

[6] Réka Albert and Albert-László Barabási. Dynamics of complex systems: Scaling laws for the period of boolean networks. *Physical Review Letters*, 84(24):5660, 2000.

[7] Réka Albert and Albert-László Barabási. Topology of evolving networks: local events and universality. *Physical review letters*, 85(24):5234, 2000.

[8] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[9] Joan M Aldous and Robin J Wilson. *Graphs and applications: an introductory approach*, volume 1. Springer Science &amp; Business Media, 2003.

[10] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, 2014.

[11] Taher Alzahrani and KJ Horadam. Community detection in bipartite networks: Algorithms and case studies. In *Complex Systems and Networks*, pages 25–50. Springer, 2016.

[12] Luıs A Nunes Amaral, Antonio Scala, Marc Barthelemy, and H Eugene Stanley. Classes of small-world networks. *Proceedings of the national academy of sciences*, 97(21):11149–11152, 2000.

[13] F Amblard. Which ties to choose? a survey of social networks models for agent-based social simulations. In *Proceedings of the 2002 SCS International Conference On Artificial Intelligence, Simulation and Planning in High Autonomy Systems*, pages 253–258, 2002.

[14] Babak Amiri, Liaquat Hossain, John W Crawford, and Rolf T Wigand. Community detection in complex networks: Multi–objective enhanced firefly algorithm. *Knowledge-Based Systems*, 46:1–11, 2013.

[15] T Antal and PL Krapivsky. Weight-driven growing networks. *Physical Review E*, 71(2):026103, 2005.

[16] Alex Arenas, Albert Díaz-Guilera, Jurgen Kurths, Yamir Moreno, and Changsong Zhou. Synchronization in complex networks. *Physics reports*, 469(3):93–153, 2008.

[17] Alex Arenas, Albert Diaz-Guilera, and Conrad J Pérez-Vicente. Synchronization reveals topological scales in complex networks. *Physical review letters*, 96(11):114102, 2006.

[18] Ery Arias-Castro, Nicolas Verzelen, et al. Community detection in dense random networks. *The Annals of Statistics*, 42(3):940–969, 2014.

[19] Martin Atzmueller, Stephan Doerfel, and Folke Mitzlaff. Description-oriented community detection using exhaustive subgroup discovery. *Information Sciences*, 329:965–984, 2016.

[20] Korinna Bade and Andreas Nürnberger. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 13–24. SIAM, 2008.

[21] Bart Baesens, Veronique Van Vlasselaer, and Wouter Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley and amp, Sons, 2015.

[22] James P Bagrow and Erik M Bollt. Local method for detecting communities. *Physical Review E*, 72(4):046108, 2005.

[23] Eric Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[24] Eric Bair and Robert Tibshirani. Semi-supervised methods to predict patient survival from gene expression data. *PLoS biology*, 2(4):e108, 2004.

[25] FA Baker, David L Verbyla, CS Hodges Jr, and EW Ross. Classification and regression tree analysis for assessing hazard of pine mortality caused by heterobasidion annosum. *Plant Disease*, 1993.

[26] VK Balakrishnan. *Schaum's Outline of Graph Theory: Including Hundreds of Solved Problems.* McGraw Hill Professional, 1997.

[27] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 11–18, 2003.

[28] Albert-László Barabási. Network science. *Phil. Trans. R. Soc. A*, 371(1987):20120375, 2013.

[29] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[30] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1):173–187, 1999.

[31] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: statistical mechanics and its applications*, 281(1):69–77, 2000.

[32] Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3):590–614, 2002.

[33] Michael J Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.

[34] Andrew D Barbour and Gesine Reinert. Small worlds. *Random Structures &amp; Algorithms*, 19(1):54–74, 2001.

[35] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks.* Cambridge university press, 2008.

[36] Alain Barrat and Martin Weigt. On the properties of small-world network models. *The European Physical Journal B-Condensed Matter and Complex Systems*, 13(3):547–560, 2000.

[37] Rodrigo C Barros, André CPLF de Carvalho, and Alex A Freitas. *Automatic design of decision-tree induction algorithms*. Springer, 2015.

[38] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[39] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004.

[40] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999.

[41] Stephen J Beckett. Improved community detection in weighted bipartite networks. *Royal Society open science*, 3(1):140536, 2016.

[42] Punam Bedi and Chhavi Sharma. Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(3):115–135, 2016.

[43] Johannes Berg and Michael Lässig. Correlated random networks. *Physical review letters*, 89(22):228701, 2002.

[44] Ginestra Bianconi and A-L Barabási. Competition and multiscaling in evolving networks. *EPL (Europhysics Letters)*, 54(4):436, 2001.

[45] Ginestra Bianconi and Albert-László Barabási. Bose-einstein condensation in complex networks. *Physical review letters*, 86(24):5632, 2001.

[46] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

[47] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[48] S Boccaletti, M Ivanchenko, V Latora, A Pluchino, and A Rapisarda. Detecting complex network modularity by dynamical clustering. *Physical Review E*, 75(4):045102, 2007.

[49] Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

[50] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[51] Marián Boguná and Romualdo Pastor-Satorras. Epidemic spreading in correlated complex networks. *Physical Review E*, 66(4):047104, 2002.

[52] Marián Boguná, Romualdo Pastor-Satorras, and Alessandro Vespignani. Epidemic spreading in complex networks with degree correlations. *arXiv preprint cond-mat/0301149*, 2003.

[53] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science &amp; Business Media, 2013.

[54] JA Bondy and US Murty. Graph theory with application, 1976.

[55] Stephen P Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005.

[56] Stephen P Borgatti and Daniel S Halgin. On network theory. *Organization science*, 22(5):1168–1181, 2011.

[57] Simone Borra and Agostino Di Ciaccio. Improving nonparametric regression methods by bagging and boosting. *Computational Statistics &amp; Data Analysis*, 38(4):407–420, 2002.

[58] Elizabeth Bott. Family and social network: Roles. *Norms, and External Relationships in Ordinary Urban Families, London: Tavistock Publications*, 1957.

[59] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[60] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[61] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees.* Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[62] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[63] Carla E Brodley and Paul E Utgoff. *Multivariate versus univariate decision trees.* University of Massachusetts, Department of Computer and Information Science Amherst, MA, 1992.

[64] Carla E Brodley and Paul E Utgoff. Multivariate decision trees. *Machine learning*, 19(1):45–77, 1995.

[65] Horst Bunke, Peter J Dickinson, Miro Kraetzl, and Walter D Wallis. *A graph-theoretic approach to enterprise network dynamics*, volume 24. Springer Science &amp; Business Media, 2007.

[66] Guido Caldarelli, Andrea Capocci, Paolo De Los Rios, and Miguel A Munoz. Scale-free networks from varying vertex intrinsic fitness. *Physical review letters*, 89(25):258702, 2002.

[67] Guido Caldarelli, Romualdo Pastor-Satorras, and Alessandro Vespignani. Structure of cycles and local ordering in complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):183–186, 2004.

[68] Fazli Can, Tansel Özyer, and Faruk Polat. *State of the art applications of social network analysis.* Springer, 2014.

[69] R Casey and George Nagy. Decision tree design using a probabilistic model (corresp.). *IEEE Transactions on Information Theory*, 30(1):93–99, 1984.

[70] Claudio Castellano, Fabio Cecconi, Vittorio Loreto, Domenico Parisi, and Filippo Radicchi. Self-contained algorithms to detect communities in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):311–319, 2004.

[71] Joe Celko. *Joe Celko's SQL Puzzles and Answers.* Morgan Kaufmann, 2006.

[72] B Chandra, Ravi Kothari, and Pallath Paul. A new node splitting measure for decision tree construction. *Pattern Recognition*, 43(8):2725–2731, 2010.

[73] Hong Chang and Dit-Yan Yeung. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 20. ACM, 2004.

[74] O Chapelle, B Schölkopf, and A Zien. Semi-supervised learning, ser. adaptive computation and machine learning, 2006.

[75] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4):1777–1787, 2012.

[76] Pin-Yu Chen and Alfred O Hero. Deep community detection. *IEEE Transactions on Signal Processing*, 63(21):5706–5719, 2015.

[77] Tianping Chen, Xiwei Liu, and Wenlian Lu. Pinning complex networks by a single controller. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(6):1317–1326, 2007.

[78] John Y. Ching, Andrew K. C. Wong, and Keith C. C. Chan. Class-dependent discretization for inductive learning from continuous and mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):641–651, 1995.

[79] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.

[80] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[81] Fan RK Chung and Ronald L Graham. *Erdos on graphs: his legacy of unsolved problems.* AK Peters Wellesley, MA, 1998.

[82] Krzysztof J Cios, Witold Pedrycz, Roman W Swiniarski, and Lukasz Andrzej Kurgan. *Data mining: a knowledge discovery approach.* Springer Science &amp; Business Media, 2007.

[83] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[84] Andrea Clementi, Miriam Di Ianni, Giorgio Gambosi, Emanuele Natale, and Riccardo Silvestri. Distributed community detection in dynamic graphs. *Theoretical Computer Science*, 584:19–41, 2015.

[85] Reuven Cohen and Shlomo Havlin. Scale-free networks are ultrasmall. *Physical review letters*, 90(5):058701, 2003.

[86] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–32, 2003.

[87] National Research Council et al. *Applications of social network analysis for building community disaster resilience: workshop summary.* National Academies Press, 2009.

[88] Jan Salomon Cramer. The origins of logistic regression. *Timbergen Institute Discussion Paper*, 2002.

[89] Nick Crossley, Christina Prell, and John Scott. Social network analysis: introduction to special edition. *Methodological Innovations Online*, 4(1):1–5, 2009.

[90] Gábor Csányi and Balázs Szendrői. Fractal–small-world dichotomy in real-world networks. *Physical Review E*, 70(1):016122, 2004.

[91] Peter Csermely. Creative elements: network-based predictions of active centres in proteins and cellular and social networks. *Trends in biochemical sciences*, 33(12):569–576, 2008.

[92] Kolaczyk Eric D and Gábor Csárdi. *Statistical analysis of network data with R*, volume 65. Springer, 2014.

[93] Pratap Dangeti. *Statistics for Machine Learning.* Packt Publishing, 2017.

[94] Jörn Davidsen, Holger Ebel, and Stefan Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 88(12):128701, 2002.

[95] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 138–149. SIAM, 2005.

[96] Guilherme F de Arruda, Luciano da Fontoura Costa, and Francisco A Rodrigues. A complex networks approach for data clustering. *Physica A: Statistical Mechanics and its Applications*, 391(23):6174–6183, 2012.

[97] Manlio De Domenico, Clara Granell, Mason A Porter, and Alex Arenas. The physics of spreading processes in multilayer networks. *Nature Physics*, 12(10):901–906, 2016.

[98] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Enhancing community detection using a network weighting strategy. *Information Sciences*, 222:648–668, 2013.

[99] JP Marques De Sá, João Gama, Raquel Sebastião, and Luís A Alexandre. Decision trees using the minimum entropy-of-error principle. In *International Conference on Computer Analysis of Images and Patterns*, pages 799–807. Springer, 2009.

[100] Glenn De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002.

[101] Glenn De'ath and Katharina E Fabricius. Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81(11):3178–3192, 2000.

[102] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[103] Reinhard Diestel. *Graph theory. Graduate texts in mathematics.* Springer-Verlag Berlin and Heidelberg GmbH amp, 2000.

[104] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[105] Thang N Dinh and My T Thai. Toward optimal community detection: From trees to general weighted networks. *Internet Mathematics*, 11(3):181–200, 2015.

[106] Luca Donetti and Miguel A Munoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, 2004.

[107] Sergey N Dorogovtsev, Alexander V Goltsev, and José FF Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275, 2008.

[108] Sergey N Dorogovtsev and JFF Mendes. Accelerated growth of networks. *arXiv preprint cond-mat/0204102*, 2002.

[109] Sergey N Dorogovtsev and Jose FF Mendes. Evolution of networks. *Advances in physics*, 51(4):1079–1187, 2002.

[110] SN Dorogovtsev and JFF Mendes. Evolution of networks: From biological nets to the internet and www. Technical report, Oxford University Press, 2003.

[111] Bruce A Draper, Carla E Brodley, and Paul E Utgoff. Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):888–893, 1994.

[112] Haifeng Du, Marcus W Feldman, Shuzhuo Li, and Xiaoyi Jin. An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity*, 12(3):53–60, 2007.

[113] Jennifer A Dunne, Richard J Williams, and Neo D Martinez. Network structure and biodiversity loss in food webs: robustness increases with connectance. *Ecology letters*, 5(4):558–567, 2002.

[114] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[115] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Physical review E*, 66(3):035103, 2002.

[116] Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the national academy of sciences*, 99(9):5825–5829, 2002.

[117] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[118] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):476–491, 1997.

[119] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[120] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

[121] De Faria et al. Application of information theory to sequential fault diagnosis. *IEEE Transactions on Computers*, 100(2):164–170, 1982.

[122] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[123] Ronald A Fisher. The statistical utilization of multiple measurements. *Annals of Human Genetics*, 8(4):376–386, 1938.

[124] Gary William Flake, Steve Lawrence, C Lee Giles, and Frans M Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, 2002.

[125] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[126] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[127] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical review E*, 70(5):056104, 2004.

[128] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[129] Jerome H Friedman. A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers*, 4(C-26):404–408, 1977.

[130] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[131] Jerome H Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics, 2001.

[132] Jing Gao, Pang-Ning Tan, and Haibin Cheng. Semi-supervised clustering with partial background information. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 489–493. SIAM, 2006.

[133] Jose A García, Rosa Rodriguez-Sánchez, and Joaquín Fdez-Valdivia. Social impact of scholarly articles in a citation network. *Journal of the Association for Information Science and Technology*, 66(1):117–127, 2015.

[134] Eugene Garfield. Citation indexes for science. *Science*, 122:108–111, 1955.

[135] Eugene Garfield et al. Science citation index-a new dimension in indexing. *Science*, 144(3619):649–654, 1964.

[136] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium auctore Carolo Friderico Gauss.* sumtibus Frid. Perthes et IH Besser, 1809.

[137] Sheila Gaynor and Eric Bair. Identification of relevant subtypes via preweighted sparse clustering. *arXiv preprint arXiv:1304.3760*, 2013.

[138] Saul B Gelfand, CS Ravishankar, and Edward J Delp. An iterative growing and pruning algorithm for classification tree design. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 818–823. IEEE, 1989.

[139] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[140] Inmar Givoni and Brendan Frey. Semi-supervised affinity propagation with instance-level constraints. In *Artificial Intelligence and Statistics*, pages 161–168, 2009.

[141] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[142] Leo A Goodman and William H Kruskal. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49(268):732–764, 1954.

[143] Steve Gregory. An algorithm to find overlapping community structure in networks. *Knowledge discovery in databases: PKDD 2007*, pages 91–102, 2007.

[144] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *Review of Machine Learning Techniques for Processing Multimedia Content*, 1:9–16, 2004.

[145] Roger Guimerà, Marta Sales-Pardo, and Luís A Nunes Amaral. Module identification in bipartite and directed networks. *Physical Review E*, 76(3):036102, 2007.

[146] Wenlong Hang, Kup-Sze Choi, Shitong Wang, and Pengjiang Qian. Semi-supervised learning using hidden feature augmentation. *Applied Soft Computing*, 59:448–461, 2017.

[147] C Hartmann, Pramod Varshney, Kishan Mehrotra, and C Gerberich. Application of information theory to the construction of efficient decision trees. *IEEE Transactions on Information Theory*, 28(4):565–577, 1982.

[148] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–310, 1986.

[149] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007, 1993.

[150] Michaela Hoffman, Douglas Steinley, Kathleen M Gates, Mitchell J Prinstein, and Michael J Brusco. Detecting clusters/communities in social networks. *Multivariate behavioral research*, pages 1–17, 2017.

[151] Petter Holme, Mikael Huss, and Hawoong Jeong. Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19(4):532–538, 2003.

[152] Erik Holmström, Nicolas Bock, and Johan Brännlund. Modularity density of network community divisions. *Physica D: Nonlinear Phenomena*, 238(14):1161–1167, 2009.

[153] Xuezhen Hong, Jun Wang, and Guande Qi. Comparison of semi-supervised and supervised approaches for classification of e-nose datasets: Case studies of tomato juices. *Chemometrics and Intelligent Laboratory Systems*, 146:457–463, 2015.

[154] Darko Hric, Richard K Darst, and Santo Fortunato. Community detection in networks: Structural communities versus ground truth. *Physical Review E*, 90(6):062805, 2014.

[155] Andrea Isoni. *Machine Learning for the Web*. Packt Publishing Ltd, 2016.

[156] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[157] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[158] Lucas GS Jeub, Prakash Balachandran, Mason A Porter, Peter J Mucha, and Michael W Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015.

[159] Sanjeev Kumar Jha, Honghan Zhao, Fitsum M Woldemeskel, and Bellie Sivakumar. Network theory and spatial rainfall connections: An interpretation. *Journal of Hydrology*, 527:13–19, 2015.

[160] Pall F Jonsson, Tamara Cavanna, Daniel Zicha, and Paul A Bates. Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. *BMC bioinformatics*, 7(1):2, 2006.

[161] Mihyun Kang and Zdenek Petrávsek. Random graphs: Theory and applications from nature to society to the brain. *Internationale Mathematische Nachrichten*, (227):1–24, 2014.

[162] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[163] Gordon V Kass. An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, pages 119–127, 1980.

[164] Nancy Katz, David Lazer, Holly Arrow, and Noshir Contractor. Network theory and small groups. *Small group research*, 35(3):307–332, 2004.

[165] Dror Y Kenett, Matjaž Perc, and Stefano Boccaletti. Networks of networks-an introduction. *Chaos, Solitons &amp; Fractals*, 80:1–6, 2015.

[166] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[167] Hans A Kestler, Johann M Kraus, Günther Palm, and Friedhelm Schwenker. On the effects of constraints in semi-supervised hierarchical clustering. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 57–66. Springer, 2006.

[168] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[169] Dan Klein, Sepandar D Kamvar, and Christopher D Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford, 2002.

[170] David Knoke and Song Yang. *Social network analysis*, volume 154. Sage, 2008.

[171] Devin C Koestler, Carmen J Marsit, Brock C Christensen, Margaret R Karagas, Raphael Bueno, David J Sugarbaker, Karl T Kelsey, and E Andres Houseman. Semi-supervised recursively partitioned mixture models for identifying cancer subtypes. *Bioinformatics*, 26(20):2578–2585, 2010.

[172] ED Kolascyk. Statistical analysis of network data. *SAMSI program on Complex networks. Boston university*, 2013.

[173] Jens Krause, DP Croft, and Richard James. Social network theory in the behavioural sciences: potential applications. *Behavioral Ecology and Sociobiology*, 62(1):15–27, 2007.

[174] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.

[175] David S Lamb, Joni A Downs, and Chanyoung Lee. The network k-function in context: examining the effects of network structure on the network k-function. *Transactions in GIS*, 20(3):448–460, 2016.

[176] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[177] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[178] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[179] Piotr Lasek. C-nbc: Neighborhood-based clustering with constraints. In *CS&amp;P*, pages 113–120. Citeseer, 2014.

[180] Martin HC Law, Alexander Topchy, and Anil K Jain. Model-based clustering with probabilistic constraints. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 641–645. SIAM, 2005.

[181] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[182] Sune Lehmann and Lars Kai Hansen. Deterministic modularity optimization. *The European Physical Journal B-Condensed Matter and Complex Systems*, 60(1):83–88, 2007.

[183] Elizabeth A Leicht and Mark EJ Newman. Community structure in directed networks. *Physical review letters*, 100(11):118703, 2008.

[184] Elizabeth Leon, Olfa Nasraoui, and Jonatan Gomez. Anomaly detection based on unsupervised niche clustering with application to network intrusion detection. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 502–508. IEEE, 2004.

[185] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[186] Cheng Li and Bingyu Wang. Fisher linear discriminant analysis, 2017.

[187] Xiang Li, Yao Wu, Martin Ester, Ben Kao, Xin Wang, and Yudian Zheng. Semi-supervised clustering in attributed heterogeneous information networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1621–1629. International World Wide Web Conferences Steering Committee, 2017.

[188] Xiaobo Li and Richard C Dubes. Tree classifier design with a permutation statistic. *Pattern Recognition*, 19(3):229–235, 1986.

[189] Kung-Yee Liang. *Generalized Linear Models Estimating Functions and Multivariate Extensions.* Institude of Statistical Science, Academia Sinica, 1999.

[190] Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348, 2014.

[191] Bo Long, Xiaoyun Xu, Zhongfei Zhang, and S Yu Philip. Community learning by graph approximation. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 232–241. IEEE, 2007.

[192] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[193] Zhengdong Lu and Todd K Leen. Semi-supervised clustering with pairwise constraints: A discriminative approach. *Journal of Machine Learning Research*, 2:299–306, 2007.

[194] David Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(Suppl 2):S186–S188, 2003.

[195] Daniel MN Maia, João EM de Oliveira, Marcos G Quiles, and Elbert EN Macau. Community detection in complex networks via adapted kuramoto dynamics. *Communications in Nonlinear Science and Numerical Simulation*, 53:130–141, 2017.

[196] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook.* Springer, 2005.

[197] Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.

[198] Yohann Mansiaux and Fabrice Carrat. Detection of independent associations in a large epidemiologic dataset: a comparison of random forests, boosted regression trees, conventional and penalized logistic regression for identifying independent factors associated with h1n1pdm influenza infections. *BMC medical research methodology*, 14(1):99, 2014.

[199] John J McArdle and Gilbert Ritschard. *Contemporary issues in exploratory data mining in the behavioral sciences*. Routledge, 2013.

[200] Volodymyr Melnykov, Igor Melnykov, and Semhar Michael. Semi-supervised model-based clustering with positive and negative constraints. *Advances in data analysis and classification*, 10(3):327–349, 2016.

[201] Mansfield Merriman. On the history of the method of least squares. *The Analyst*, 4(2):33–36, 1877.

[202] Stanley Milgram. The small world problem. *Psychology Today,*, pages 60–67, 1967.

[203] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3(4):319–342, 1989.

[204] Mehdi Rezapoor Mirsaleh and Mohammad Reza Meybodi. A michigan memetic algorithm for solving the community detection problem in complex network. *Neurocomputing*, 214:535–545, 2016.

[205] James Clyde Mitchell. *Social networks in urban situations: analyses of personal relationships in Central African towns*. Manchester University Press, 1969.

[206] Sadaaki Miyamoto and Akihisa Terami. Semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[207] Francesco Mola and Roberta Siciliano. A fast splitting procedure for classification trees. *Statistics and Computing*, 7(3):209–216, 1997.

[208] Seyed Ahmad Moosavi, Mehrdad Jalali, Negin Misaghian, Shahaboddin Shamshirband, and Mohammad Hossein Anisi. Community detection in social networks using user frequent pattern mining. *Knowledge and Information Systems*, 51(1):159–186, 2017.

[209] Jacob L Moreno. *Who shall survive*, volume 58. JSTOR, 1934.

[210] James N Morgan. History and potential of binary segmentation for exploratory data analysis. *Journal of Data Science*, 3(2):123–136, 2005.

[211] Kevin P MURPHY. Machine learning: a probabilistic perspective, 2012.

[212] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[213] Sreerama K Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32, 1994.

[214] Dávid Natingga. *Data Science Algorithms in a Week*. Packt Publishing, 2017.

[215] J. A. NELDER and R. W. M. WEDDERBURN. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(1):370–384, 1972.

[216] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[217] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[218] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[219] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[220] Mark EJ Newman. Analysis of weighted networks. *Physical review E*, 70(5):056131, 2004.

[221] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[222] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[223] Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[224] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[225] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[226] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[227] Nam P Nguyen, Thang N Dinh, Yilin Shen, and My T Thai. Dynamic social community detection and its applications. *PloS one*, 9(4):e91431, 2014.

[228] Bruno Magalhães Nogueira, Yuri Karan Benevides Tomas, and Ricardo Marcondes Marcacini. Integrating distance metric learning and cluster-level constraints in semi-supervised clustering. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 4118–4125. IEEE, 2017.

[229] Georgios Paliouras, Symeon Papadopoulos, and Dimitrios Vogiatzis. Discovery of complex user communities. In *User Community Discovery*, pages 1–22. Springer, 2015.

[230] Georgios Paliouras, Symeon Papadopoulos, Dimitrios Vogiatzis, and Yiannis Kompatsiaris. *User Community Discovery*. Springer, 2015.

[231] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.

[232] David Papo, Javier M Buldú, Stefano Boccaletti, and Edward T Bullmore. Complex network theory and the brain, 2014.

[233] DL Passmore. Social network analysis: Theory and applications (2014). *Tersedia: http://code. pediapress. com/[12 Juni 2014]*, 2016.

[234] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[235] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65(3):036104, 2002.

[236] Peter Pollner, Gergely Palla, and Tamas Vicsek. Preferential attachment of communities: The same principle, but a higher level. *EPL (Europhysics Letters)*, 73(3):478, 2005.

[237] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.

[238] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[239] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. *ICASE LaRC Interdisciplinary Series in Science and Engineering*, 4:323–368, 1997.

[240] Josep M Pujol, Javier Béjar, and Jordi Delgado. Clustering algorithm for determining community structure in large networks. *Physical Review E*, 74(1):016107, 2006.

[241] Xingqin Qi, Wenliang Tang, Yezhou Wu, Guodong Guo, Eddie Fuller, and Cun-Quan Zhang. Optimal local community detection in social networks based on density drop of subgraphs. *Pattern Recognition Letters*, 36:46–53, 2014.

[242] Guoqi Qian, Yuehua Wu, Davide Ferrari, Puxue Qiao, and Frédéric Hollande. Semisupervised clustering by iterative partition and regression with neuroscience applications. *Computational intelligence and neuroscience*, 2016, 2016.

[243] F Questier, R Put, D Coomans, B Walczak, and Y Vander Heyden. The use of cart and multivariate regression trees for supervised and unsupervised feature selection. *Chemometrics and Intelligent Laboratory Systems*, 76(1):45–54, 2005.

[244] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[245] Filippo Radicchi. A paradox in community detection. *EPL (Europhysics Letters)*, 106(3):38001, 2014.

[246] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[247] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[248] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[249] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

[250] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[251] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[252] Thomas Richardson, Peter J Mucha, and Mason A Porter. Spectral tripartitioning of networks. *Physical Review E*, 80(3):036111, 2009.

[253] Gilbert Ritschard. Chaid and earlier supervised tree methods, 2010.

[254] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[255] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[256] EM Rounds. A combined nonparametric approach to feature selection and binary decision tree design. *Pattern Recognition*, 12(5):313–317, 1980.

[257] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 216–223. Springer, 2007.

[258] Mohammad Ebrahim Samie and Ali Hamzeh. Community detection in dynamic social networks: A local evolutionary approach. *Journal of Information Science*, page 0165551516657717, 2016.

[259] Bilal Saoud and Abdelouahab Moussaoui. Community detection in networks based on minimum spanning tree and modularity. *Physica A: Statistical Mechanics and its Applications*, 460:230–234, 2016.

[260] Marialisa Scatà, Alessandro Di Stefano, Aurelio La Corte, Pietro Liò, Emanuele Catania, Ermanno Guardo, and Salvatore Pagano. Combining evolutionary game theory and network theory to analyze human cooperation patterns. *Chaos, Solitons &amp; Fractals*, 91:17–24, 2016.

[261] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[262] John Scott. *Social network analysis.* Sage, 2017.

[263] John P Scott. *Social Network Analysis: A Handbook.* SAGE Publications Sage UK: London, England, 2000.

[264] Ishwar Krishnan Sethi and GPR Sarvarayudu. Hierarchical classifier design using mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 4(4):441–445, 1982.

[265] Ronghua Shang, Jing Bai, Licheng Jiao, and Chao Jin. Community detection based on modularity and an improved genetic algorithm. *Physica A: Statistical Mechanics and its Applications*, 392(5):1215–1231, 2013.

[266] CE Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[267] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. In *Advances in neural information processing systems*, pages 465–472, 2004.

[268] Chuan Shi, Yanan Cai, Di Fu, Yuxiao Dong, and Bin Wu. A link clustering based overlapping community detection algorithm. *Data &amp; Knowledge Engineering*, 87:394–404, 2013.

[269] Thiago Christiano Silva and Liang Zhao. *Machine learning in complex networks*, volume 1. Springer, 2016.

[270] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the Association for Information Science and Technology*, 24(4):265–269, 1973.

[271] Chaoming Song, Shlomo Havlin, and Hernan A Makse. Self-similarity of complex networks. *Nature*, 433(7024):392–395, 2005.

[272] Harold W Sorenson. Least-squares estimation: from gauss to kalman. *IEEE spectrum*, 7(7):63–68, 1970.

[273] Niko Speybroeck. Classification and regression trees. *International journal of public health*, 57(1):243–246, 2012.

[274] Dan Steinberg. Cart: classification and regression trees. *The top ten algorithms in data mining*, 9:179, 2009.

[275] Stephen M Stigler. Gauss and the invention of least squares. *The Annals of Statistics*, pages 465–474, 1981.

[276] Carolin Strobl, James Malley, and Gerhard Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4):323, 2009.

[277] Steven H Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.

[278] Chang Su, Yue Yu, Xianzhong Xie, and Yukun Wang. Data sensitive recommendation based on community detection. *Foundations of Computing and Decision Sciences*, 40(2):143–159, 2015.

[279] Peng Gang Sun. Community detection by fuzzy clustering. *Physica A: Statistical Mechanics and its Applications*, 419:408–416, 2015.

[280] Peng Gang Sun. Imbalance problem in community detection. *Physica A: Statistical Mechanics and its Applications*, 457:364–376, 2016.

[281] Peng Gang Sun and Xiya Sun. Complete graph model for community detection. *Physica A: Statistical Mechanics and its Applications*, 471:88–97, 2017.

[282] Zhaocai Sun, Yunming Ye, Xiaofeng Zhang, Zhexue Huang, Shudong Chen, and Zhi Liu. Batch-mode active learning with semi-supervised cluster tree for text classification. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 388–395. IEEE Computer Society, 2012.

[283] Clifton D Sutton. Classification and regression trees, bagging, and boosting. *Handbook of statistics*, 24:303–329, 2005.

[284] Jan L Talmon. A multiclass nonparametric partitioning algorithm. *Pattern Recognition Letters*, 4(1):31–38, 1986.

[285] Wei Tang, Hui Xiong, Shi Zhong, and Jie Wu. Enhancing semi-supervised clustering: a feature projection perspective. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 707–716. ACM, 2007.

[286] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002.

[287] Bojun Tu, Zhihua Zhang, Shusen Wang, and Hui Qian. Making fisher discriminant analysis scalable. In *International Conference on Machine Learning*, pages 964–972, 2014.

[288] Paul E Utgo and Carla E Brodley. Linear machine decision trees. Technical report, Tech. Rep. COINS 91-10, University of Massachusetts, Amherst, MA, USA, 1991.

[289] Diego Vallejo-Huanga, Paulina Morillo, and Cesar Ferri. Semi-supervised clustering algorithms for grouping scientific articles. *Procedia Computer Science*, 108:325–334, 2017.

[290] Joost van Loon. Network. *Theory, Culture  Society*, 23((2–3)):307–314, 2006.

[291] I Vragović and E Louis. Network community structure and loop coefficient method. *Physical Review E*, 74(1):016105, 2006.

[292] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097:577–584, 2000.

[293] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[294] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276. ACM, 2007.

[295] Walter D Wallis. *A beginner's guide to graph theory.* Springer Science &amp; Business Media, 2010.

[296] Gaoxia Wang, Yi Shen, and Ming Ouyang. A vector partitioning approach to detecting community structure in complex networks. *Computers &amp; Mathematics with Applications*, 55(12):2746–2752, 2008.

[297] Jiaoe Wang, Huihui Mo, Fahui Wang, and Fengjun Jin. Exploring the network structure and nodal centrality of china's air transport network: A complex network approach. *Journal of Transport Geography*, 19(4):712–721, 2011.

[298] Meng Wang, Chaokun Wang, Jeffrey Xu Yu, and Jun Zhang. Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment*, 8(10):998–1009, 2015.

[299] Qin Wang, Guangping Zeng, and Xuyan Tu. Information technology project portfolio implementation process optimization based on complex network theory and entropy. *Entropy*, 19(6):287, 2017.

[300] Christopher P Warren, Leonard M Sander, and Igor M Sokolov. Geography in a scale-free network model. *Physical Review E*, 66(5):056105, 2002.

[301] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[302] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[303] Allan P White and Wei Zhong Liu. Bias in information-based measures in decision tree induction. *Machine Learning*, 15(3):321–329, 1994.

[304] Leland Wilkinson. *Classification and regression trees.* 2004.

[305] Rick L Wilson and Peter A Rosen. Protecting data through perturbation techniques: The impact on knowledge discovery in databases. *Journal of Database Management (JDM)*, 14(2):14–26, 2003.

[306] Andrew Y Wu, Michael Garland, and Jiawei Han. Mining scale-free networks using geodesic clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 719–724. ACM, 2004.

[307] Biao Xiang, En-Hong Chen, and Tao Zhou. Finding community structure based on subgraph similarity. *Complex Networks*, pages 73–81, 2009.

[308] Gao Xiang and Wang Min. Applying semi-supervised cluster algorithm for anomaly detection. In *Information Processing (ISIP), 2010 Third International Symposium on*, pages 43–45. IEEE, 2010.

[309] Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612, 2008.

[310] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):43, 2013.

[311] Zheng Xie, Zhenzheng Ouyang, Pengyuan Zhang, Dongyun Yi, and Dexing Kong. Modeling the citation network by network cosmology. *PloS one*, 10(3):e0120687, 2015.

[312] Xin Xin, Chaokun Wang, Xiang Ying, and Boyang Wang. Deep community detection in topologically incomplete networks. *Physica A: Statistical Mechanics and its Applications*, 469:342–352, 2017.

[313] Gang Xu, Sophia Tsoka, and Lazaros G Papageorgiou. Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B-Condensed Matter and Complex Systems*, 60(2):231–239, 2007.

[314] Hao Xu, Yanli Hu, Zhenwen Wang, Jianwei Ma, and Weidong Xiao. Core-based dynamic community detection in mobile social networks. *Entropy*, 15(12):5419–5438, 2013.

[315] Nai-Ru Xu, Jia-Bao Liu, De-Xun Li, and Jun Wang. Research on evolutionary mechanism of agile supply chain network via complex network theory. *Mathematical Problems in Engineering*, 2016, 2016.

[316] Ramon Xulvi-Brunet and Igor M Sokolov. Reshuffling scale-free networks: From random to assortative. *Physical Review E*, 70(6):066102, 2004.

[317] Yang Yang, Peng Gang Sun, Xia Hu, and Zhou Jun Li. Closed walks for community detection. *Physica A: Statistical Mechanics and its Applications*, 397:129–143, 2014.

[318] Luh Yen, Francois Fouss, Christine Decaestecker, Pascal Francq, and Marco Saerens. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data &amp; Knowledge Engineering*, 68(3):338–361, 2009.

[319] Jinfeng Yi, Lijun Zhang, Tianbao Yang, Wei Liu, and Jun Wang. An efficient semi-supervised clustering algorithm with sequential constraints. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1405–1414. ACM, 2015.

[320] Yisehac Yohannes and John Hoddinott. Classification and regression trees: an introduction. *International Food Policy Research Institute*, 2033, 1999.

[321] Soon-Hyung Yook, Hawoong Jeong, A-L Barabási, and Yuhai Tu. Weighted evolving networks. *Physical review letters*, 86(25):5835, 2001.

[322] Tetsuya Yoshida. A graph-based approach for semisupervised clustering. *Computational Intelligence*, 30(2):263–284, 2014.

[323] Mina Zarei and Keivan Aghababaei Samani. Eigenvectors of network complement reveal community structure more accurately. *Physica A: Statistical Mechanics and its Applications*, 388(8):1721–1730, 2009.

[324] Heping Zhang and Yuanqing Ye. A tree-based method for modeling a multivariate ordinal response. *Statistics and its interface*, 1(1):169, 2008.

[325] Peng Zhang, Jinliang Wang, Xiaojia Li, Menghui Li, Zengru Di, and Ying Fan. Clustering coefficient and community structure of bipartite networks. *Physica A: Statistical Mechanics and its Applications*, 387(27):6869–6875, 2008.

[326] Li Zheng and Tao Li. Semi-supervised hierarchical clustering. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 982–991. IEEE, 2011.

[327] Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical review e*, 67(6):061901, 2003.

[328] Jiang ZHU, Bai WANG, Bin WU, and Weiyu ZHANG. Emotional community detection in social network. *IEICE Transactions on Information and Systems*, 100(10):2515–2525, 2017.

[329] Mu Zhu, Fanrong Meng, and Yong Zhou. Semisupervised clustering for networks based on fast affinity propagation. *Mathematical Problems in Engineering*, 2013, 2013.

[330] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.