

Optimization of the REED web application for automatic outlier detection in predictive medicine

Dissertation presented by
Sue LAMBOT , Aurélien PINOLET

for obtaining the Master's degree in
Computer Science and Engineering

Supervisor(s)
Pierre DUPONT

Reader(s)
Jérôme PAUL, Roberto D'AMBROSIO

Academic year 2016-2017

Abstract

This master thesis aims at proposing a solution to improve the current outlier detection method used by the REED online tool developed by DNAlytics and performing a preliminary - fully automated - analysis of uploaded medical datasets in order to determine if the data exhibits enough interest to justify the costs of starting a real, more in depth, manual analysis.

An outlier being a sample for which the behavior indicates that it might not come from the same distribution as the normal observation, it has an impact on the analysis that can be made on the datasets and it is therefore often needed to identify those points in order to handle them correctly. Detecting outliers, especially on medical datasets is not trivial due to the non-normal behavior of some of the features, the very high dimensionality, and the small number of samples usually available. This work will explore the different techniques found in the literature and try to find a solution that is both robust and more efficient than the currently implemented one.

The different experiments performed using public datasets (considering the minority class as outliers) or simulated datasets (with the last percentile of the distribution as outliers).

The usage of Ensemble methods, even if they are usually not performing as well as the best base detectors shows a stabilizing factor which tends to suggest that using ensemble is a really good solution in an unsupervised environment requiring robustness such as REED.

Moreover, the improvement offered by the bagging seems really promising, as expected due to the strong impact of the curse of dimensionality.

Unfortunately, most of our experiments are based on synthetic datasets due to the lack of good unsupervised assessment methods. The literature does not offer any real leads on how to tackle the lack of ground truth, preventing to draw any strong conclusions from the unsupervised experiments.

We would like to thank

our supervisor, Professor Pierre Dupont for the kind directions he gave us when we struggled on some concepts;

Jérôme Paul for the abundant advises he gave us;

the people from DNAlytics: Pierre Gramme, Thibault Helleputte and Jérôme Paul for their support and guidance throughout the year;

and Professor Pierre Dupont, Jérôme Paul and Roberto D'Ambrosio for the times they will dedicate to the reading of this document.

Contents

1	Introduction	1
1.1	DNAlytics and REED	1
1.1.1	Current outlier detection	1
1.1.2	Constraints caused by the automation	2
1.2	What is an outlier?	2
1.2.1	Outliers' origin	3
1.2.2	Different types of outliers	3
1.3	Peculiarities of REED's datasets and of biomedical data	4
1.4	Our work	5
2	State of the art: outlier detection	7
2.1	Input and Output types	7
2.1.1	Data type	7
2.1.2	Data Labels	7
2.1.3	Output of Outlier Detection	8
2.2	Detection techniques	9
2.2.1	Probabilistic and Statistical Modelling	9
2.2.2	Linear Models	10
2.2.3	Proximity based techniques	13
2.2.4	Clustering based	14
2.2.5	Distance-based approaches	18
2.2.6	Density-based approaches	18
2.2.7	Ensemble methods	24
2.3	Curse of Dimensionality	26
2.3.1	Dimensionality Reduction	27
3	Implementation	29
3.1	Code structure	29
3.2	Pre-processing	29
3.2.1	Data transformations	29
3.2.2	Dimensionality reduction	31
3.3	Methods	31
3.3.1	Two phases clustering	31
3.3.2	Hierarchical Clustering	32
3.3.3	Local correlation integral	34
3.3.4	Local outlier factor	34
3.3.5	One-class SVM	35
3.3.6	PCA	35
3.3.7	Ensemble	35

4	Evaluation protocol	37
4.1	Different evaluation protocols seen in the literature	37
4.1.1	Datasets	37
4.1.2	Validation method	38
4.2	Validation protocol	38
4.2.1	Supervised	38
4.2.2	Unsupervised	39
4.2.3	Benchmark	42
4.3	Statistical test	43
4.4	Visualization	45
4.4.1	Boxplot of the scores	45
4.4.2	PCA projection	45
5	Experiments	47
5.1	Experimental setup	47
5.1.1	Public datasets	47
5.1.2	Synthetic datasets	48
5.2	Dataset transformation	50
5.2.1	Normalization	50
5.2.2	Dimensionality reduction	51
5.2.3	Pre-processing by detector	53
5.3	Ensemble	54
5.3.1	Base detector selection	54
5.3.2	Internal function	55
5.3.3	Base detector weight determination	56
5.3.4	Global comparison	56
5.4	Supervised experiments	57
5.4.1	Base detectors	57
5.4.2	Unsupervised	67
5.5	Scores distributions	69
5.6	Computational performances	70
6	Results interpretation	73
6.1	Dataset transformation	73
6.1.1	Normalization	73
6.1.2	Dimensionality reduction	74
6.1.3	Per detector analysis	74
6.1.4	Conclusion	75
6.2	Ensemble	75
6.2.1	Base detectors comparison	76
6.2.2	Internal functions comparison	76
6.2.3	Base detectors weights	76
6.2.4	Ensemble: conclusion	77
6.3	Supervised results	77
6.3.1	public files	77
6.3.2	Synthetic files	78
6.4	Unsupervised	79
6.5	Threshold discussion - Visualization of the scores distributions	81
6.6	Analysis conclusion	81
6.6.1	Base detectors	81
6.6.2	Ensemble	82

7 Conclusion	83
Appendices	85
A Additional figures	87
A.1 Preprocessing by method for public datasets	88
A.2 Boxplot visualisation of the scores	90
Bibliographic references	92

Table of notations

General notations

α	a proportion parameter between 0 and 1
$\text{RVE}_{\hat{t}}(\mathbf{e})$	A robust variance estimator based on the eigenvectors $\mathbf{e}$ and a subset of $\hat{t}$ samples
$\hat{d}$	the number of PCA components in HRPCA
$\hat{t}$	a lower bound on the number of authentic points in HRPCA
λ	an eigenvalue
Σ	a correlation matrix
$\mathbf{D}$	a distance matrix
$\mathbf{I}$	the identity matrix
$\mathbf{L}$	a low rank matrix
$\mathbf{M}$	a matrix representing the PCA's input data
$\mathbf{S}$	a sparse matrix
$\mathbf{X}$	the data matrix
$\bar{T}$	an upper bound on the number of removed sample in HRPCA
Θ	a decision threshold
μ	a vector of means
$\mathbf{c}$	a set of centroïds in K-means
$\mathbf{e}$	an eigenvector
score	a vector of scores for each data point

$\mathbf{w}$	a weight vector
$\mathbf{x}$	a vector of features for a data instance
$\mathbf{y}$	a boolean vector indicating the outlier status of the instances
i	a specific data instance index
j	a specific feature instance index
Q_1, Q_3	the first and third quartiles
a, b, c	some constants
B	the number of base detectors
C	a set of cluster
d	the number of dimensions
E	a set of edges
F	a forest
IQR	the interquartile range
J	an objective function
K	a number of clusters
L	a list of detectors
p	a number of sub-tree/Clusters
r	distance used in the r-neighborhood
T	a tree
t	a number of samples
V	a set of vertices

Chapter 1

Introduction

This master thesis aims at proposing a solution to improve the current outlier detection method used by the REED application by DNAlytics. An outlier being a point for which the behavior indicates that it might not come from the same distribution as the normal points, it has an impact on the analysis that can be made on the datasets and it is therefore often needed to identify those points in order to handle them correctly. Detecting outliers, especially on medical datasets is not trivial due to the non-normal behavior of some of the features, the very high dimensionality, and the small number of samples usually available. This work will explore the different techniques found in the literature and try to find a solution that is both robust and more efficient than the currently implemented one.

This thesis has been structured in different chapter: Chapter 2 will first approach the problem from a theoretical point of view and will describe the different direction taken by outlier detection in the literature, Chapter 4 will lay out the evaluation protocol that will be used to perform our experiments, our implementation will be quickly explained in Chapter 3 before presenting the results in Chapter 5. We will finish with a discussion on the results in Chapter 6 and a conclusion in Chapter 7.

Before diving deeper into the subject, this first chapter will explain the context in which this work will be used (Section 1.1), define more precisely what an outlier is (Section 1.2), expose the peculiarities of the biomedical data (Section 1.3) and introduce our work in Section 1.4.

1.1 DNAlytics and REED

DNAlytics is a spin-off from of Université Catholique de Louvain providing expert services and products in the field of machine learning dedicated to data-driven personalized medicine. One of those products is REED, which stands for Rapid and Easy Evaluation of Datasets. It is a tool that has been developed by the company to help them prospect for new data mining projects.

REED is an online tool performing a preliminary - fully automated - analysis of an uploaded dataset in order to determine if its data exhibit enough interest to justify the costs of starting a real, more in depth, manual analysis. The tool provides customers with relevant information relative to their dataset, such as the amount of constants, significant variables, transformed variables together with an indication of the potential performances and the number of outliers detected in the dataset.

This last metric is the one of most interest in this thesis.

1.1.1 Current outlier detection

The current software already implements an outlier detection technique but it only considers the features independently as if they were not correlated and flags as outliers the points having a certain percentage of features showing an abnormal deviation.

1.1.2 Constraints caused by the automation

The REED software works in a one fits all approach with no optimizations based on the dataset. Additionally, even though most of the datasets uploaded on the software exhibit certain similarities (explained in 1.3) the customers are free to upload any kind of datasets which means that no assumption at all can be made on the type of the data that will need to be handled.

The technique used for the outlier detection therefore needs to be robust and to work for different types of input features. The solution needs to:

- work for numerical and categorical features;
- handle a very variable number of dimensions, going from a few to several thousand;
- work without requiring manual parameters selection.

Lots of techniques[28] rely on first training a classifier based on labeled samples to define what the normal behavior is and then flag as outliers the samples that deviate from this normal behavior. For this application, as we don't have any labeled data. We cannot use those techniques. All the techniques that rely on continuous learning are also no applicable as we only evaluate the data once. For the same reason, live streaming techniques are not applicable either.

1.2 What is an outlier?

An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs. [17][16]

An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.[17][5]

Outlier detection aims to find patterns in data that do not conform to expected behavior.[28]

Each article offers its own vision, which are, if somehow close, always slightly different. The fact that so many different, and rather vague, definitions coexist is closely related to what makes outlier detection so difficult: outliers are not defined by what they are but by what they are not, normal points. Because of this, referring to a ground truth in a real, large, dataset is often impossible as trying to define what an outlier is by describing its behavior would somehow reduce the scope to known outliers only.

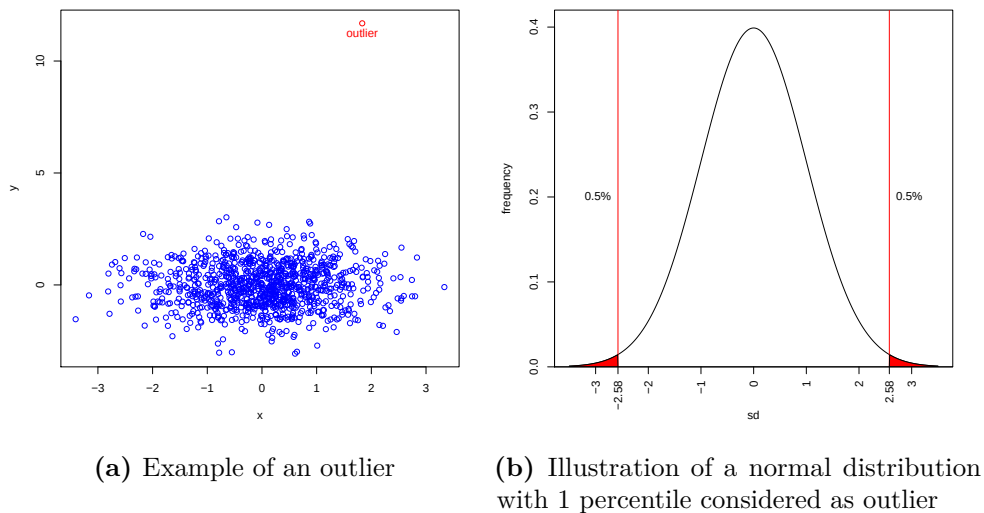


Figure 1.1: Outlier representation

In order to avoid placing expectations on what behaviors the outliers should have, we will therefore, in this study, focus on the basic, closer to a mathematical definition, idea that outliers are samples that do not appear to be drawn from the same distribution as normal, inliers, points. This vision is illustrated in figure 1.1 where the points considered as having an outlier behavior are those located past the last percentile of the distribution.

1.2.0.1 Noise

On top of clear outliers, the data can also include varying levels of noise which can make the outlier detection more difficult as their behavior does not stand out as visibly than in a non-noisy dataset. As illustrated in figure 1.2.

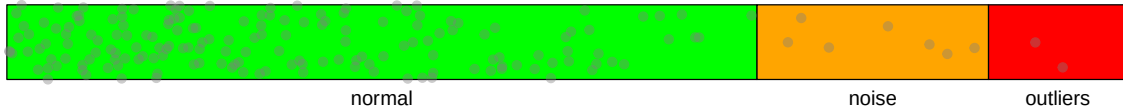


Figure 1.2: Illustration of the presence of noise

1.2.1 Outliers' origin

The presence of outliers in a dataset can have different sources, identified in [17] as :

- mechanical faults or instrument error;
- human mistake;
- natural deviation in the population.
- surprising veridical data: a point that belongs to one class but behaves like the other.

1.2.2 Different types of outliers

[10] and [28] identifies three types of outliers:

A point (or global) outlier is an individual sample that is anomalous compared to the rest of the data, such as in figure 1.1a.

A contextual outlier is an individual sample that is anomalous in a specific context.

The contextual attributes are the attributes determining the context for the instances.

The behavioral attributes are the attributes that are considered as context-dependent.

The fact that a sample is an outlier, or not, is determined based on the value of the behavioral attributes within their specific context.

For example, negative temperatures are anomalous in summer but normal in winter. A feature defining the season would therefore be a contextual attribute while the temperature would be a behavioral attribute.

Collective outliers are samples in a series that together express an anomalous behavior compared to the rest of the data. The samples might not be outliers individually, but their occurrence together is. Those types of outliers are found in data having a "between samples" relationship, such as sequence data, graph data and spacial data.

It's to be noted that, while point outliers and collective outliers require different detection techniques, they can both be contextual outliers if they are analysed with respect to a context. This means the detection techniques for collective and point outliers can be used to detect the corresponding contextual outliers by incorporating a context.

1.3 Peculiarities of REED's datasets and of biomedical data

The aim of this section is to better understand the peculiarities of the datasets likely to be uploaded on the application in order to perform a better selection of the detection techniques to consider. The following description of the biomedical datasets is mainly coming from the PhD thesis of Jérôme Paul[25], our main contact at DNAlytics as it corresponds to the guidelines we received from DNAlytics during our meetings with them.

The datasets often contain per patient observations. Those observations are composed of different types of features corresponding to different biomedical measurements. The nature of the measurements can vary and come from different sources:

- medical equipment which typically measure a very high number of variables, for example:
 - microarray genomic screening which estimates the level of activity of genes.
 - flow cytometry analysis which evaluates the concentration of proteins present in a sample.
- medical or environmental factors manually encoded by physicians such as the body temperature, the blood pressure, family history, ...

Those datasets can thus be composed of a mixture of continuous, distinct and categorical variables, and, as all the measurements may not be present for each sample, the datasets may have missing values for some observations. Samples are also usually very expensive; therefore, the datasets really often contain only a very small number of observations with a high number of features. The datasets can also be designed for classification tasks (for example: responders/non-responders to a treatment) or for regression.

DNAlytics also strongly urged that normality cannot be assumed as

- the nature of the features is unknown;
- the gene expressions are typically not normally distributed but often follow a distribution that is heavily right-skewed. An example of such a distribution is visible on figure 1.3.

REED is therefore trying, as much as possible, not to use parametric methods.

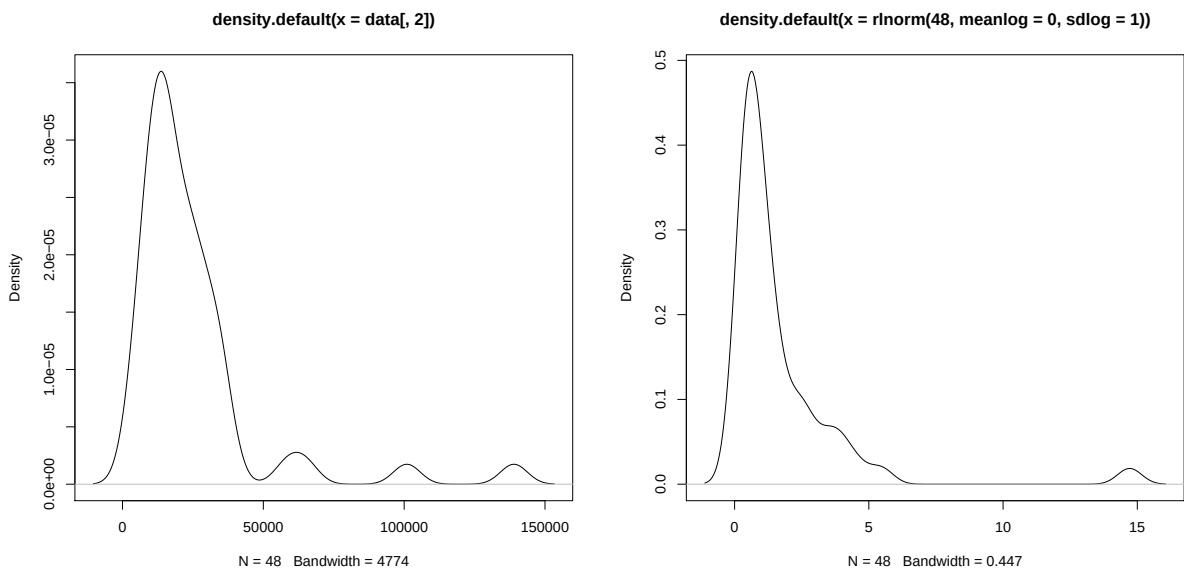


Figure 1.3: Example of right-skewed feature distribution from real, anonymized, datasets received from DNAlytics and of a log-normal distribution of the same number of samples

1.4 Our work

In this perspective, our work will focus on finding a solution for global outliers and - for classification problems - for contextual outliers considering the class as the contextual attribute as the feature distributions may be different for samples belonging to different classes.

The datasets submitted on REED typically being really small; the solution will have to focus on robustness to reduce the amount of false positive detection. It will also have to be non-dependent of optimization parameters and be able to handle various feature types and a very variable number of dimensions going to just a few to several thousand as the nature of the uploaded dataset is totally unknown. Ideally, the detection method should be multivariate to take advantage of the correlation between the features, which is not handled by the detection method currently implemented.

Our main angle of approach will be to use techniques from different families as base detectors and an ensemble method, performing a prediction based on the returned results from the methods it is composed of. This technique should reduce the false positive caused by techniques unfit to handle some datasets and therefore offer an overall more robust prediction.

In the next chapter, we will investigate the different families of detection techniques that exist in order to determine which ones are applicable to our specific problem.

Chapter 2

State of the art: outlier detection

This chapter will focus on exploring the different techniques found in the literature that might be relevant to our problem.

We will start by reminding the possible types of input and output that can be given and produced by an outlier detection algorithm in Section 2.1 to better narrow the scope of the techniques to investigate and will follow with the description of the different techniques we have envisioned in Section 2.2.

2.1 Input and Output types

2.1.1 Data type

Continuous variables are variables which can take an infinite set of uncountable values. For example, a set of values from $\mathbb{R}$. The values of a continuous variable can be sorted in a meaningful way; there is a relation between the different values that the variable can take.

Discrete variables can only take a finite set of infinite set of countable values. For example, a set of values from $\mathbb{N}$.

Boolean variables are discrete variables which are limited to the set of values $\{\text{TRUE}(1), \text{FALSE}(0)\}$.

Categorical variables are discrete variables for which the value cannot be ordered in a meaningful way (or at least, not without knowing the meaning of categories). For example, $\{\text{male}, \text{female}\}$.

2.1.2 Data Labels

The survey performed by Victoria J Hodge and Jim Austin in [17] separates the different outlier detection techniques in three categories based on the availability of labels to train the models:

Type 1: unsupervised No prior knowledge of the data is used by those methods, they are typically defining which samples are outliers based on their divergence with the majority of the samples.

Type 2: supervised Those methods model both normality and abnormality and therefore require data tagged as normal or abnormal for training.

Type 3: semi-supervised Those methods model only the normality and try to define a boundary around the normal samples. They require a training set containing only normal samples. The abnormality is recognized as diverging from this normality.

The task we have to perform corresponds to unsupervised outlier detection. As we don't know anything about the original distribution, determining which point is or is not an outlier requires some assumptions over the data distribution. We will assume that:

- most of the data conforms to the expected behavior and that outliers are the data differing from the norm;
- as most of the data have a similar behavior, the outliers are located in sparser area;
- as most of the data have a similar behavior, the distance between the outliers and the rest of the data is higher.

Some of the semi-supervised detection methods might also be studied under the assumption that most of the samples are normal.

2.1.3 Output of Outlier Detection

The detection techniques can either return a score or a binary decision.

Scores The scoring techniques assign an outlier score to each sample. Defining which sample is or is not an outlier thus requires to define a threshold on the scores.

Labels Labels determining which points are normal or outliers can be assigned to each sample.

Scoring has several advantages over a simple outlier decision. [28] pointed out that they allow to define the threshold more precisely based on domain specific information. This cannot be directly applied in this case as we cannot fine-tune the parameters for the datasets but they, however, offer some benefits as they can be seen as a kind of one-dimensional projection of the dataset representing how different the samples are from each other. It's a fit measure of the samples to the model of normal data.

- they provide more information on how confident the detector is on the fact that a specific sample is, or is not, an outlier based on how skewed the score distribution is;
- one-dimensional extreme value analysis can be used to determine the threshold based on the score distribution as outliers tend to be distant from the normal points;
- they allow to define a threshold based on criteria such as trying to reduce the false positives;
- they can be combined via ensemble methods to form aggregated new scores as we will see in Section 2.2.7;
- scores can be converted to probability estimates as proposed in [15] by using the Expectation-maximization (EM) algorithm¹ under the assumption that the normal points follow an exponential distribution while the outliers follow a gaussian distribution as, according to [1], this is the typical behavior of outliers scores.

As a side note, if scores can be converted into binary decision by setting up a threshold, binary decision can also be converted into scores using, for example, a score of 0 for normal points and a score of 1 for outliers as illustrated in equations 2.1 and 2.2 with Θ as threshold parameter and the binary vector $\mathbf{y}$ containing **TRUE** for outliers and **FALSE** for normal points.

$$y_i = \text{score}_i \geq \Theta \quad (2.1)$$

$$\text{score}_i = \begin{cases} 1 & \text{if } y_i = \text{TRUE} \\ 0 & \text{if } y_i = \text{FALSE} \end{cases} \quad (2.2)$$

¹The EM algorithm is an iterative algorithm, used to find the maximum likelihood of some parameters.

The output type is therefore not a determining criterion to decide which techniques to use or not, but scores will be kept and not converted into binary decision when available as they convey more information.

2.2 Detection techniques

[17] and [1] insist on the importance of selecting an algorithm that can accurately model the data distribution in order to highlight the outlying points. This might seem obvious but, in the specific problem we are dealing with; this would mean selecting an algorithm that is accurate for any kind of encountered dataset. Finding an optimal algorithm doing such a thing is, of course, not possible. What we are going to do in this section is to see, in the different families of algorithms, which ones are able to handle the kind of datasets that are the most frequently encountered by REED: mainly numerical data with a high number of dimensions and a small number of samples.

In this section, the general descriptions of the main families of techniques are coming from the Outlier Analysis book by Charu Aggarwal[1] unless otherwise specified. The sources for the more specific, detailed, methods will be cited when introducing them.

2.2.1 Probabilistic and Statistical Modelling

Most probabilistic methods are trying to fit the data to a parametric model representing their distributions and are highly dependent on the specific choices of data distribution with which the modeling is performed. As we don't have any insight on the shape of the data distribution we are dealing with in REED, most of those techniques are not applicable and will therefore not be explored. We will only mention a few basic methods that have been used during our work. Mainly extreme value analysis which can be used for the evaluation of the outliers scores and the Mahalanobis distance.

Strength can be used for any data type as long as a generative model is available for each mixture component.

Weakness as they try to fit the value to a specific distribution, they can easily over fit and model the outliers together with the normal points if there are too many model parameters. Moreover, These methods are not easy to interpret.

2.2.1.1 The Mahalanobis distance

Using the Mahalanobis distance as an outlier score can in principle be really effective as it uses the correlation between the variables thanks to the Σ matrix and is parameter-free, but it assumes the normality of the variables and requires having a good estimate of μ and Σ . Those values are influenced by the presence of outliers and require a dataset of a good size to be estimated. The general equation works with two different points, but in this context, the second point, will be the mean vector, μ .

$$\text{Mahalanobis}(\mathbf{x}, \mu, \Sigma) = \sqrt{(\mathbf{x} - \mu)\Sigma^{-1}(\mathbf{x} - \mu)^T} \quad (2.3)$$

Interestingly, there is a relation between the Mahalanobis distance and the Euclidean distance. The Euclidean distance is not considering the correlation between the variables, it only considers the distance between two points (in this case, a sample and the means of the distribution) as visible in equation 2.4. For this reason, it is really important to rescale the variables before

computing the Euclidean distance while it is not necessary for the Mahalanobis distance, the different variables scales being considered thanks to the Σ matrix.

$$\text{Euclidean}(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T} = \sqrt{(\mathbf{x} - \mu)\mathbf{I}(\mathbf{x} - \mu)^T} \quad (2.4)$$

The Euclidean distance is therefore equal to the Mahalanobis distance if Σ is equal to the identity matrix, which means, for uncorrelated rescaled variables.

This is interesting as it can be used, as discussed in 2.2.2.1 with a robust PCA to compute scores based on a projection that reduces the impact of the outliers.

2.2.1.2 Gaussian Mixtures

Mixture models can be seen as probabilistic versions of clustering algorithms that learns the parameters defining the samples' distribution instead of the shape of their boundaries. A model composed of Gaussian mixtures can be learned using the EM algorithm. Hypothetically, the EM algorithm can be used to fit any kind of distribution but requires to know the shape of the distribution, which, is not known in our case.

2.2.1.3 Extreme value analysis

Extreme value analysis has an important role in outlier detection as outliers usually exhibits an extreme behavior. It cannot be used directly to detect some outliers when the samples are coming from a mixture of distribution - outliers between the clusters are not considered as extreme even when they are isolated - but it can be used, to some extends by almost all the detection algorithms as, even when it cannot be applied directly to the original data, those techniques can be used to analyze the scores returned by the outlier detectors.

Distance to the Interquartile range (Boxplot) The detector currently implemented in REED uses univariate extreme value analysis on the variables to evaluate which samples are outliers.

As for the boxplot, this detector is based on a distance of the data points to the interquartile range (representing 50% of the data and computed based on equation 2.5). It then considers as outliers the points which have a certain percentage of their features located at a certain distance of this range.

$$IQR = Q_3 - Q_1 \quad (2.5)$$

$$\text{scoring}(x_i^j, Q_{1,j}, Q_{3,j}) = \begin{cases} Q_{1,j} - 1.5 * IQR_j - x_i^j & \text{if } x_i^j < Q_{1,j} - 1.5 * IQR_j \\ x_i^j - Q_{3,j} + 1.5 * IQR_j & \text{if } x_i^j > Q_{3,j} + 1.5 * IQR_j \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

This method makes assumptions about the data distribution as the distance from this interquartile range at which a point can be considered as an outlier depends on the shape of the distribution. For example, in equations 2.6, the distance 1.5 is the one used to represent the outer whisker of the boxplot and is a sensible value for normally distributed data.

This method is really robust to false positive as the detected outliers are only the ones for which an abnormal behavior can be detected at the feature level on several features. It, however, does not take advantage of the feature's correlation and can therefore miss outliers.

2.2.2 Linear Models

The models presented in this section are, models, based on a linear separation but includes methods working with kernels allowing projections in a modified space.

2.2.2.1 PCA

A PCA is a statistical procedure allowing to reduce the number of variables by projecting the data onto a new axis system capturing most of their variance. This projection allows to convert a set of possibly correlated variables into a set of linearly uncorrelated variables. Each component captures the largest possible remaining variance under the constraint that it is orthogonal to the preceding components. They are computed by eigenvalue decomposition of the data covariance matrix (Σ) usually after mean centering and data matrix normalization. An example of projections is present on figure 2.1.

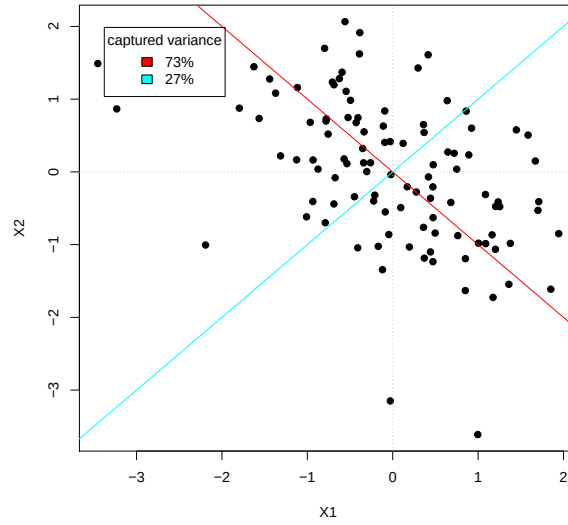


Figure 2.1: representation of the PCA rotation axes in the original space

The axis system defined by those eigenvectors allows to represent/approximate the original data with very little loss.

Property 2.1. PCA Properties[1] *Principal component analysis provides a set of eigenvectors satisfying the following properties:*

- *if the data is transformed to the axis-system corresponding to the orthogonal eigenvectors, the variance of the transformed data along each axis (eigenvector) is equal to the corresponding eigenvalue. The covariances of the transformed data in the new representation are 0;*
- *since the variance of the transformed data along the eigenvector with small eigenvalues are low, significant deviations of the transformed data from the mean values along these directions may represent outliers.*

The outlier score can thus be computed as the normalized distance of the data point to the centroid along the direction of each principal component, equation 2.7. The eigenvalue λ_j (variance) in the denominator provides a soft weighting of the distance to the different components. This score is equivalent to the Mahalanobis distance computed on the original space in Subsection 2.2.1.1

$$\text{scoring}(\mathbf{x}) = \sum_{j=1}^d \frac{|(\mathbf{x} - \mu)\mathbf{e}_j|^2}{\lambda_j} \quad (2.7)$$

2.2.2.2 Robust PCA

As the outliers are used as well as the normal data to construct the PCA, it is generally sensitive to the presence of outliers. One way to overcome this problem is to use robust PCA instead of the simple PCA.

We explored two different techniques leading to similar results:

The first one uses a matrix decomposition of the input data, $\mathbf{M}$, into a low-rank, $\mathbf{L}$, and sparse, $\mathbf{S}$, matrices (equation 2.8). [9]

$$\mathbf{M} = \mathbf{L} + \mathbf{S} \quad (2.8)$$

This can be archived by solving the following minimization problem (Principal Component Pursuit)

$$\begin{aligned} & \text{minimize } \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \\ & \text{subject to } \mathbf{L} + \mathbf{S} = \mathbf{M} \end{aligned}$$

with $\|\mathbf{M}\|_* := \sum_i \sigma_i(\mathbf{M})$, the nuclear norm of the matrix $\mathbf{M}$ and $\|\mathbf{M}\|_1 = \sum_{ij} |\mathbf{M}_{ij}|$ the l_1 -norm of the matrix $\mathbf{M}$.

The second one, using the HRPCA[30] algorithm (algorithm 1) which is designed to work in high-dimensional space. It iteratively removes (with a certain probability) the most distant points and keep the PCA which maximizes the Robust Variance Estimator (RVE) visible in equation 2.9.

$$\text{RVE}_{\hat{t}}(\mathbf{e}) = \frac{1}{\hat{t}} \sum_{i=1}^{\hat{t}} |\mathbf{e}^T \mathbf{x}_{(i)}|^2 \quad (2.9)$$

The algorithm is using a probability to remove points based on their distance instead of just removing the most distant point to prevent algorithm to be driven to a suboptimal solution.

Algorithm 1: HR-PCA

input : Contaminated sample-set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^p, \hat{d}, \bar{T}, \hat{t}$.

output : $\bar{\mathbf{e}}_1, \dots, \bar{\mathbf{e}}_{\hat{d}}$.

Let $\hat{\mathbf{x}}_i := \mathbf{x}_i$ for $i = 1, \dots, n$; $\hat{\mathbf{X}} := \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n\}$; $s := 0$; $Opt := 0$.

while $s \leq \bar{T}$ **do**

 Compute the empirical variance matrix :

$$\hat{\Sigma} := \frac{1}{n-s} \sum_{i=1}^{n-s} \hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^T$$

 Perform PCA on $\hat{\Sigma}$. Let $\mathbf{e}_1, \dots, \mathbf{e}_{\hat{d}}$ be the $\hat{d}$ principal components of $\hat{\Sigma}$.

if $\sum_{j=1}^{\hat{d}} \text{RVE}_{\hat{t}}(\mathbf{e}_j) > Opt$ **then**

 let $Opt := \sum_{j=1}^{\hat{d}} \text{RVE}_{\hat{t}}(\mathbf{e}_j)$ and let $\bar{\mathbf{e}}_j := \mathbf{e}_j$ for $j=1, \dots, \hat{d}$.

end

 Randomly remove a point from $\{\hat{\mathbf{x}}_i\}_{i=1}^{n-s}$ according to :

$$Pr(\hat{\mathbf{x}}_i \text{ is removed from } \hat{\mathbf{X}}) \propto \sum_{j=1}^{\hat{d}} (\mathbf{e}_j^T \hat{\mathbf{x}}_i)^2$$

 Denote the remaining points by $\{\hat{\mathbf{x}}_i\}_{i=1}^{n-s-1}$

$s := s+1$

end

With $\hat{t}$, a lower bound on the number of authentic samples which is used to compute the robust variance estimator, $\hat{d}$ the number of components to use in the PCA and $\hat{T}$ an upper bound on the number of removed samples.

2.2.2.3 One-class Support Vector Machine

Similarly to the PCA projection method, the one-class SVM method builds a model considering that it is fed with samples belonging to the normal class and can then be used to discover future samples not belonging to that class depending on which side of the decision vector they are on.

It's meant to be a semi-supervised learning technique, given samples from the normal class, be able to detect abnormal samples due to the slack variables and the soft margins. When fed with a dataset containing outliers, the resulting model will be noisy but can still be used to evaluate which samples are more likely to be outliers, as long as the dataset contains a majority of samples belonging to the normal class.

It is based on the assumption that the origin of a kernel-based transformed representation belongs to the outlier class. The mapping from $\mathbf{x}$ to this kernel space $\phi(\mathbf{x})$ is not done explicitly but it will be performed implicitly within the optimization problem .

In 2.10, J is the objective function, $\mathbf{w}$ is the SVM weight vectors, b is the margin and c is a constant that regulates the trade off between false positives and false negatives.

$$\text{Minimize } J = \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{\text{Regularizer}} + \underbrace{\frac{c}{n} \sum_{i=1}^n \max\{b - \mathbf{w} \cdot \phi(\mathbf{x}_i, 0)\}}_{\text{Training Data Penalty}} - \underbrace{b}_{\text{Origin Reward}} \quad (2.10)$$

The optimization problem tries to classify as many as possible of the training points to be positive. The negative points are the detected outliers.

Their distance to the decision boundary can be used as their outlier score

The downside of the one-class SVM method is that it can be really sensitive to the choice of kernel and to the different model parameters. It is therefore ill suited for applications where we don't know the nature of the data.

2.2.3 Proximity based techniques

Proximity based methods are simple to implement and have the benefit of not making any assumption over the data distribution model [17]. They can be used to detect global outliers as well as contextual outliers as they are based on the similarities between the samples and can therefore detect outliers even when the distribution has multiple modes.

The proximity based techniques are based on the notion of distance or of density. Those two notions are highly similar as the density depends on the distance between the samples.

Depending on the algorithm and of how the problem is expressed, the outliers are considered as:

- points located in the **less dense** areas
- points with the **greatest distance** to the others
- points which are the **most distant** from the **clusters' center**

Figure 2.2 illustrates this with two distinct clusters of points and a lone outlier.

Strength They do not try to model the distribution.

Weakness As they are based on the computation of distances between all the records; their complexity is directly proportional to both the dimensionality and the number of records. They can easily suffer from the curse of dimensionality.[17]

Those methods will be developed more in detail in the following section. Even though clustering algorithms can be based on either density or distance, they have been grouped in one section as they share the same base concept.

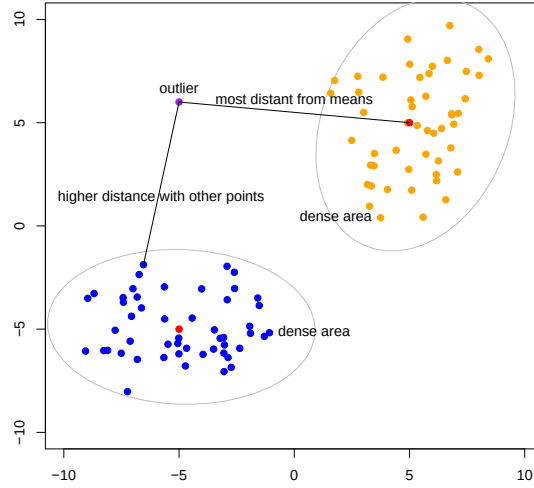


Figure 2.2: Two clusters of data with the different notion of distance and proximity illustrated

2.2.4 Clustering based

Cluster analysis is defined as the process of splitting a set of objects into multiple clusters, such that members of one cluster look similar. The goal of this partitioning may be to gain an insight into some structure in the population. As points with peculiar behavior are more likely to be isolated in small clusters, clustering approach can be really useful in outlier detection.

But as stated in [17], as the clustering algorithms are typically based on the distance between the normal and abnormal samples, the presence of noise in the dataset can have a negative impact on the performance of clustering techniques.

2.2.4.1 K-means

The purpose of the K-means algorithm is to find a good (but not perfect) partition with a fixed, K , number of clusters. The initial step is to take K random centroids. The next step is to assign each point to its closest centroid and recompute the position of all the centroids as the center of mass of all its points. The second step is repeated until there is no move in the centroids.

Algorithm 2: K-means process

```

input :  $\mathbf{X}$  the set of input samples
 $K$  the number of clusters wanted
output :  $\mathcal{C}$  the set of clusters
 $\mathcal{C} \leftarrow K$  empty sets //  $\mathcal{C}_k$ : the set of points in cluster  $k$ 
 $c \leftarrow$  a set of  $K$  randomly chosen centroids
repeat
  // Allocating process:
  for  $k$  from 1 to  $K$  do
     $\mathcal{C}_k \leftarrow \{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{X}, k = \operatorname{argmin}_{k'}(\mathbf{x}_i, \mathbf{c}_{k'})\}$ 
  end
  // Moving process:
  for  $k$  from 1 to  $K$  do
     $\mathbf{c}_k \leftarrow \operatorname{mean}(\mathcal{C}_k)$ 
  end
until no mean change between two consecutive iterations;
return :  $\mathcal{C}$ 

```

The outliers can be identified as the point located in really small clusters, as the cluster

initialization is done randomly, multiple runs of the algorithm are usually required. The K-means algorithm is presented in algorithm 2. The computation time of K-means is $O(n)$ [17].

2.2.4.2 K-medoids

The K-medoids algorithm represents each cluster using an actual sample and a radius rather than a prototype (average) point and a radius. This method is more robust than K-means to outliers as it uses an actual sample instead of an average which can be influenced by the presence of outliers.[17]

However, K-means outperforms K-medoids and can handle larger data sets more efficiently as K-medoids can require $O(n^2)$ running time per iteration whereas K-means is $O(n)$.

There exists optimization of the K-medoids algorithm, for example, the CLARANS (Ng and Han, 1994) algorithm which improves the computational time[17] by using a pre-specified number of potential medoids and selecting the first medoid it tests which improves the cluster quality instead of searching the entire data set for the optimal one. However, it still has a $O(n^2K)$ computational complexity which prevents using it for big datasets.

2.2.4.3 Two Phases Clustering

The method described in this section is based on the paper *Two-phase clustering process for outliers detection* [18].

This clustering method considers as outliers the points isolated in small clusters far from most of the other partitions. These small groups are called *remainder clusters*. Among all the data, the proportion of items in the *remainder clusters* is unimportant and all these items are out of the ordinary compared to usual data points. The advantage of this approach, against other more classical clustering methods, is the fact that the pattern between clusters is not neglected.

As indicated by its name, this approach is cut into two phases:

1. defining clusters;
2. finding isolated small clusters.

Note that the behavior of all the points in a single cluster is assumed to be the same. This assumption leads to the fact that all the points of a cluster have the same chance of being an outlier.

First phase: Defining clusters There exists different types of classical clustering approach using **k-means**, **hierarchical clustering** or **graph clustering**:

1. **K-means algorithm** Described in Section 2.2.4.1.
2. **Hierarchical clustering method** The purpose of the hierarchical clustering method is to build a hierarchy of clusters called the *cluster tree*. The initial step is to place each sample in its own cluster. The next step is to reduce the number of clusters. To achieve this, the two closest clusters are merged and the new centroid is located at the center of mass of all the points of the cluster. The second step is repeated until only a single cluster remains. The last step is to cut the cluster tree to have the desired number of clusters.
3. **Graph clustering theory** A graph is a mathematical structure useful in cluster analysis. In this theory, the graph, that is assimilated to a tree (there is no loop), is defined by a set of vertices, V , representing clusters; and a set of edges, E , representing the interactions between them. A tree T is written as $T = (V, E)$. The assumption of taking a tree is done to greatly simplify the computation. In the case of a tree, cutting $p - 1$ edges leads to

create p sub-tree, called $T_k \forall k \in 1..p$. C_k is its vertex set. In conclusion, $C_1, \dots, C_p$ is a feasible partition of the data into p clusters. Note that, finding the best partition in a tree can be done in polynomial time with dynamic programming, but the problem is NP-hard for other types of graph.

The method kept by the article for the first step is a modified K-means presented in algorithm 3. The difference with its classical version is the fact that the number of clusters is not fixed. With this new constraint two new rules must be added:

1. **Splitting process** A cluster will be split if the distance between the two closest centroids (written $\min(\text{dist}(\mathbf{c}))$) is smaller than the distance between this centroid and its closest point (written $\min(\text{dist}(\mathbf{x}_i, \mathbf{c}))$).
2. **Merging phase** The two closest centroids will be merged if the number of clusters is greater than a threshold K_{max} .

Algorithm 3: Modified k-means process, MKP

```

input :  $\mathbf{X}$  the set of input samples
         $K_{max}$  the maximal number of clusters we may afford
         $K_{current}$  the current number of clusters
output :  $\mathbf{c}$  the set of centroids
 $\mathbf{c} \leftarrow$  a set of  $K_{current}$  randomly chosen centroids
 $\mathbf{C} \leftarrow K_{current}$  empty sets
for  $i$  from 1 to  $n$  do
    if  $\min(\text{dist}(\mathbf{x}_i, \mathbf{c})) > \min(\text{dist}(\mathbf{c}))$  then
        // splitting process
         $K_{current} \leftarrow K_{current} + 1$ 
         $\mathbf{c}_{K_{current}} \leftarrow \mathbf{x}_i$ 
        // merging process
        if  $K_{current} > K_{max}$  then
            // merge the two closest clusters  $(\mathbf{C}_{k'}, \mathbf{C}_{k''})$  into one cluster
             $\mathbf{c} \leftarrow \mathbf{c} \setminus \{\mathbf{c}_{k'}, \mathbf{c}_{k''}\}$ 
            //  $n_k$  the number of points contained in the cluster of centroid  $\mathbf{c}_k$ 
             $\mathbf{c}_{k_{max}} \leftarrow \frac{n_{k'} * \mathbf{c}_{k'} + n_{k''} * \mathbf{c}_{k''}}{n_{k'} + n_{k''}}$ 
             $K_{current} \leftarrow K_{max}$ 
        end
    end
    // Allocating process:
    for  $k$  from 1 to  $K_{current}$  do
        |  $\mathbf{C}_k \leftarrow \{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{X}, k = \text{argmin}_{k'}(\mathbf{x}_i, \mathbf{c}_{k'})\}$ 
    end
    // Moving process:
    for  $k$  from 1 to  $K_{current}$  do
        |  $\mathbf{c}_k \leftarrow \text{mean}(\mathbf{C}_k)$ 
    end
end
return :  $\mathbf{c}$ 

```

Second phase: Finding isolated small clusters The second phase is called the outlier finding process (OFP) and is highly inspired from the graph clustering theory described above. The clusters determined in the previous phase are used as the nodes of a fully connected graph. Therefore, a minimum spanning tree algorithm will be used to keep only the smallest connections between all the clusters and discard any loop. The next step is to remove the longest edge of the spanning tree to isolate remote clusters from each other. This step is repeated until the

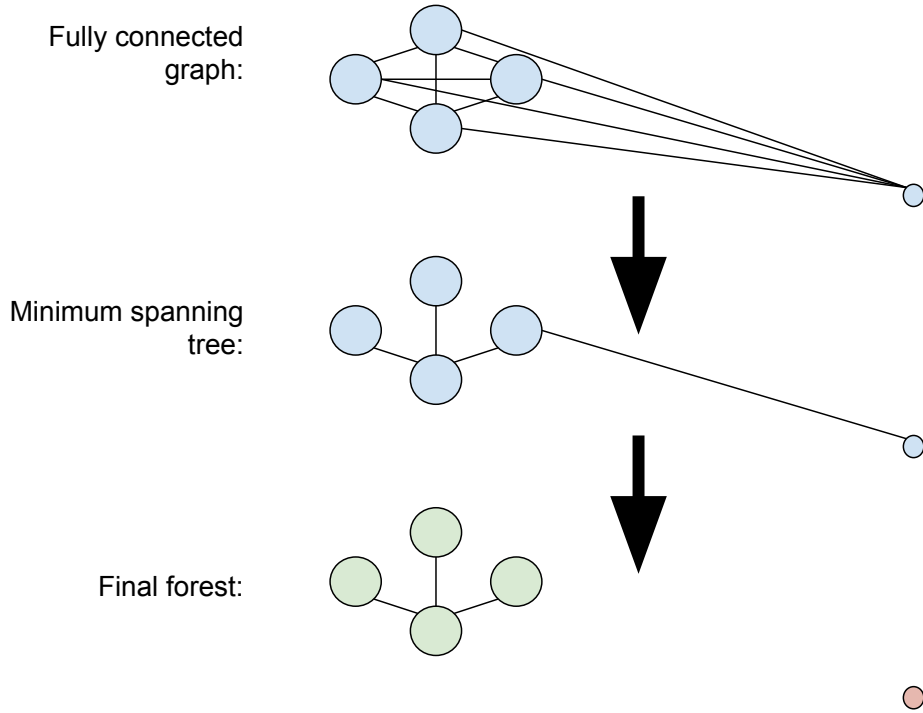


Figure 2.3: The forest construction based on minimum spanning tree

number of sub-tree in the forest is sufficiently high. The smallest sub-trees are seen as outliers. A graphical representation of the OFP is present on figure 2.3 while its algorithm can be found in algorithm 4.

Algorithm 4: OFP

```

input   :  $c$  the set of centroids
            $K$  the number of desired sub-trees
output : outliers the set of outliers
 $F \leftarrow \text{MST}(c)$ 
while  $\text{length}(F) \neq K$  do
  | // Remove the longest edge of  $F$  (cut a tree in two sub-trees)
end
// Smallest sub-trees regarded as outliers
return : Smallest sub-trees

```

2.2.4.4 Hierarchical Clustering

The following section is mainly based on the paper *Outlier Detection Using Clustering Methods: a data cleaning application*[21].

Using hierarchical clustering methods is motivated by the unbalanced distribution of outliers compared to the inliers. By definition, outliers represents a very low proportion of the total dataset. At the initialization, this fact will conduct to spread the outliers across all the clusters when using non-hierarchical clustering approach, and therefore, lead to unstable results for outlier detection.

Using this approach, we expect that outliers will be placed in the smallest clusters. The assumption behind this hypothesis is that outlier observations, having unusual values, will be

distant from the usual data and consequently be isolated in smaller clusters. Therefore, the size of the resulting clusters should be a good indicator of the presence of outliers. The bigger it is, the less likely it is to contain outliers. The hierarchical clustering algorithm for outlier detection can be found in algorithm 5.

Algorithm 5: Basic hierarchical clustering

```

input :  $\mathbf{X}$  the set of input samples
        dist a distance function
         $K$  the number of clusters
         $\Theta$  the threshold for the size of the clusters
output : outliers the set of outliers
outliers  $\leftarrow$  NULL
 $\mathbf{D} \leftarrow$  compute the distance matrix from  $\mathbf{X}$  by using dist
htree  $\leftarrow$  compute the hierarchy tree from  $\mathbf{D}$ 
 $\mathbf{C} \leftarrow$  Cut htree into  $K$  clusters
for  $k$  from 1 to  $K$  do
    //  $n_k$  the number of points contained in the cluster of centroid  $\mathbf{c}_k$ 
    if  $n_k < \Theta$  then
        | outliers  $\leftarrow$  outliers  $\cup \{\text{obs} \mid \text{obs} \in C_k\}$ 
    end
end
return : outliers

```

A critical parameter of the algorithm is the number of clusters, K , as taking:

- too many clusters will isolate inliers;
- not enough clusters will hide outliers in large clusters.

Having a good number of clusters will lead to put outliers in small clusters. The source [21] take $K = \max(2, n/10)$ as heuristic. This allows to have a dynamic number of clusters depending on the input size.

2.2.5 Distance-based approaches

2.2.5.1 K Nearest Neighbours

The K-NN approaches are based on the number of samples that can be found within a specified distance of the considered sample. If less a specified number of neighbors are found within the distance when considering the sample's K nearest neighbors, the sample is seen as an outlier. This method requires to compute the distance between each point and the other points and is therefore dependent on the number of dimensions and the number of samples. It is also susceptible to the Curse of Dimensionality (see Section 2.3)[17]

It also requires defining parameters such as the distance that is considered and the number of required neighbors.

2.2.6 Density-based approaches

The number of points in a specific region of the space is used to define outliers in density-based methods. These are really close to distance-based and clustering approach. The reason is that, distances, clusters and densities are highly connected. All these methods have for purpose to find *locality-sensitive outliers*. Nevertheless, it can exist both local and global density-based approaches.

Local versus global analysis Using global methods is often less efficient than local one as they only capture *global outliers*. Moreover, global methods can lead to erroneous results if the density has a high variability over the space.

An example of a global method could be to look at the number of neighbors in a fixed size region around a sample, such as K-NN. Now, let's imagine two clusters with the same number of objects. The first one is really dense, but the second one is not. If the size of the neighborhood is fixed, the number of neighbors in the sparse region will be much smaller than in the dense cluster. This implies that all these points have a higher chance to be outliers, as they have less neighbors. Which is inconsistent with the fact that all the points are in a cluster (of same size). Another problem could be that an outlier, outside of the two clusters, is not too far from the dense region. Therefore, this point has less chance to be an outlier than all the points in the sparse cluster, as the outlier has more neighbors for the same size area. A graphical representation of the problem is in figure 2.4.

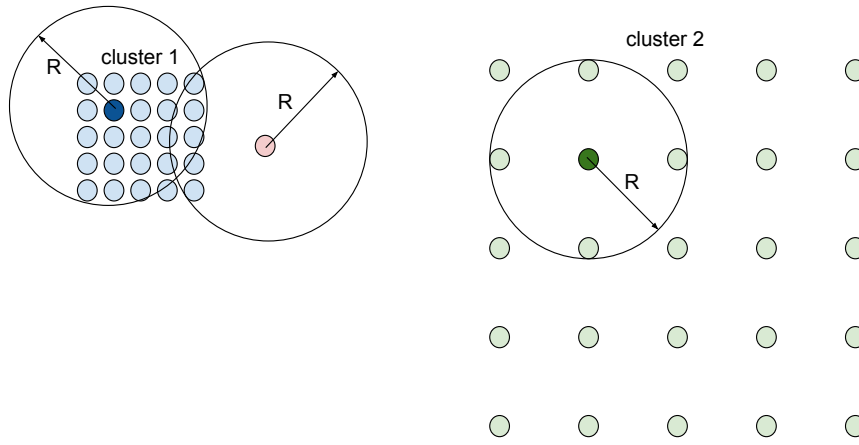


Figure 2.4: Graphical representation of one type of problem with global methods

2.2.6.1 Local outlier factor (LOF)

The following section is highly inspired by the paper *LOF: Identifying Density-Based Local Outliers*[8].

The local outlier factor can be seen as the outlierness degree of a sample, this (continuous) property is adjusted for local density variations.

Formal definitions

Definition 2.1 (k-distance). The k-distance, k-dist, of an object $\mathbf{x}_i$, is defined as the distance $\text{dist}(\mathbf{x}_i, \mathbf{x}_l)$ such that:

- for **at least** k objects $\mathbf{x}_{l'} \in \mathbf{X} \setminus \{\mathbf{x}_i\} : \text{dist}(\mathbf{x}_i, \mathbf{x}_{l'}) \leq \text{dist}(\mathbf{x}_i, \mathbf{x}_l)$
- for **at most** k-1 objects $\mathbf{x}_{l'} \in \mathbf{X} \setminus \{\mathbf{x}_i\} : \text{dist}(\mathbf{x}_i, \mathbf{x}_{l'}) < \text{dist}(\mathbf{x}_i, \mathbf{x}_l)$

Definition 2.2 (k-distance neighborhood). The k-distance neighborhood, N_k , of an object $\mathbf{x}_i$ is the distance around $\mathbf{x}_i$ that contains every object, whose distance from $\mathbf{x}_i$ is not greater than the k-distance:

$$N_k(\mathbf{x}_i) = \{\mathbf{x}_l \in \mathbf{X} \setminus \{\mathbf{x}_i\} | \text{dist}(\mathbf{x}_i, \mathbf{x}_l) \leq \text{k-dist}(\mathbf{x}_i)\}$$

Definition 2.3 (reachability distance). The reachability distance, reach-dist_k , from an object $\mathbf{x}_i$ to an object $\mathbf{x}_l$ is defined as:

$$\text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_l) = \max(\text{k-dist}(\mathbf{x}_l), \text{dist}(\mathbf{x}_i, \mathbf{x}_l))$$

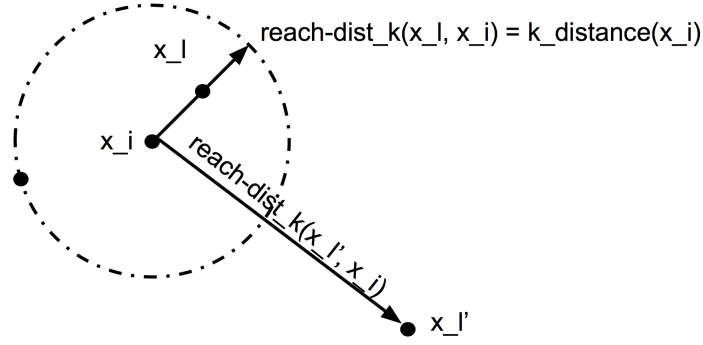


Figure 2.5: Example of the reachability distance of x_i with $k = 3$

The reason for using reachability distance instead of a classical distance method is to significantly reduce the statistical fluctuations of $\text{dist}(\mathbf{x}_i, \mathbf{x}_l)$ for all the points close to $\mathbf{x}_l$. This smoothing effect can be managed by tuning the factor k . The higher the value of k , the more similar $\text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_l)$ while varying $\mathbf{x}_i$. An important thing to notice is that the reachability distance is not symmetric: $\text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_l) \neq \text{reach-dist}_k(\mathbf{x}_l, \mathbf{x}_i)$. An example of reachability distance can be found in figure 2.5.

But for the original purpose, which is defining outliers, we must use an instantiation of k , which relies on two parameters:

- a parameter specifying the minimum number of objects, **MinPts**;
- a parameter specifying a volume.

This method is called density-based, as we compare the densities of different sets of objects.

As the computation needs to be done dynamically, **MinPts** is the only parameter kept. Indeed, the k -distance (with $k = \text{MinPts}$) can be used to define the area around $\mathbf{x}_i$ and $\text{reach-dist}_{\text{MinPts}}(\mathbf{x}_i, \mathbf{x}_l)$, (for $\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)$) can be used to determine the density in the neighborhood of $\mathbf{x}_i$.

From there, two different new definitions of the local outlier factor will be defined to show the proximity between density-based and distance-based approaches.

Density-based LOF

Definition 2.4 (local reachability density). The local reachability density, lrd , of $\mathbf{x}_i$ is defined as:

$$\text{lrd}_{\text{MinPts}}(\mathbf{x}_i) = \left(\frac{\sum_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{reach-dist}_{\text{MinPts}}(\mathbf{x}_i, \mathbf{x}_l)}{|N_{\text{MinPts}}(\mathbf{x}_i)|} \right)^{-1}$$

The local reachability density of $\mathbf{x}_i$ is the inverse of the average reachability distance-based on its **MinPts**-nearest neighbors. Note that this value could be infinite if there are more than **MinPts** samples at the same position.

Definition 2.5 (Density-based local outlier factor). The local outlier factor, lof , of $\mathbf{x}_i$ is defined as:

$$\text{lof}_{\text{MinPts}}(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \frac{\text{lrd}_{\text{MinPts}}(\mathbf{x}_l)}{\text{lrd}_{\text{MinPts}}(\mathbf{x}_i)}}{|N_{\text{MinPts}}(\mathbf{x}_i)|}$$

Distance-based LOF

Definition 2.6 (average reachability distance). The average reachability distance, $\text{av-reach-dist}_{\text{MinPts}}$, of $\mathbf{x}_i$ correspond to the average of the reachability distance of $\mathbf{x}_i$ to all its neighbors. It can be defined as:

$$\text{av-reach-dist}_{\text{MinPts}}(\mathbf{x}_i) = \text{mean}_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{reach-dist}_{\text{MinPts}}(\mathbf{x}_i, \mathbf{x}_l)$$

Note: The average reachability distance of a point is exactly equal to the inverse of its local reachability density.

The local outlier factor of a point is simply the ratio of its average reachability distance and the mean of the average reachability distance of its neighbors.

Definition 2.7 (Density-based LOF). The local outlier factor of $\mathbf{x}_i$ is defined as:

$$\text{lof}_{\text{MinPts}}(\mathbf{x}_i) = \frac{\text{av-reach-dist}_{\text{MinPts}}(\mathbf{x}_i)}{\text{mean}_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{av-reach-dist}_{\text{MinPts}}(\mathbf{x}_l)}$$

It can easily be demonstrated that the two versions of LOF are equivalent:

$$\begin{aligned} \text{lof}_{\text{MinPts}}(\mathbf{x}_i) &= \frac{\sum_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \frac{\text{lrd}_{\text{MinPts}}(\mathbf{x}_l)}{\text{lrd}_{\text{MinPts}}(\mathbf{x}_i)}}{|N_{\text{MinPts}}(\mathbf{x}_i)|} \\ &= \frac{\sum_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{lrd}_{\text{MinPts}}(\mathbf{x}_l)}{|N_{\text{MinPts}}(\mathbf{x}_i)|} * \frac{1}{\text{lrd}_{\text{MinPts}}(\mathbf{x}_i)} \\ &= \frac{\text{mean}_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{lrd}_{\text{MinPts}}(\mathbf{x}_l)}{\text{lrd}_{\text{MinPts}}(\mathbf{x}_i)} \\ &= \frac{\text{av-reach-dist}_{\text{MinPts}}(\mathbf{x}_i)}{\text{mean}_{\mathbf{x}_l \in N_{\text{MinPts}}(\mathbf{x}_i)} \text{av-reach-dist}_{\text{MinPts}}(\mathbf{x}_l)} \end{aligned}$$

With these definitions, the local outlier factor can't be expressed as a *pure* density-based approach as a density should be a number of points in a given area. In that case, the definition is inverted as we defined an area in which there is a given number of samples. To be more accurate, LOF is a distance-based approach with smoothing.

The local outlier factor of $\mathbf{x}_i$ captures its degree of outlierness. It is the average of the ratio of the local reachability density of $\mathbf{x}_i$ and those of its $\mathbf{x}_l$ MinPts-nearest neighbors. It is easy to see that a higher LOF is encountered when the local reachability density of $\mathbf{x}_i$ is small and the mean local reachability density of its neighborhood is high.

Moreover the local outlier factor is an accurate measure to detect points outside clusters.

Lemma 2.8 (LOF for Objects Deep in a Cluster). Let C a cluster, $\text{reach-dist-min/max}$ the minimum/maximum reachability distance of objects in C , $\text{reach-dist-min/max} = \min/\max(\text{reach-dist}(\mathbf{x}_i, \mathbf{x}_l) | \mathbf{x}_i, \mathbf{x}_l \in C)$ and $\epsilon := \frac{\text{reach-dist-max}}{\text{reach-dist-min}} - 1$. $\forall \mathbf{x}_i \in C$:

1. all the MinPts-nearest neighbors $\mathbf{x}_l$ of $\mathbf{x}_i$ are in C
2. all the MinPts-nearest neighbors $\mathbf{x}_k$ of $\mathbf{x}_l$ are in C

This leads to the conclusion² that $\text{LOF}(\mathbf{x}_i) \in [(1 + \epsilon)^{-1}; (1 + \epsilon)]$. The LOF value of any point in a cluster will be really close to one, independently of the spread of the cluster.

LOF algorithm The algorithm to compute the local outlier factor for each point in a dataset is on algorithm 6.

²The sketch of the proof is present in [8]

Algorithm 6: LOF ranking algorithm

```

input   :  $\mathbf{X}$  the set of input samples
            $MinPts$  Number of points in the neighborhood
output :  $\mathbf{score}$  a vector of score for each sample
 $\mathbf{score} \leftarrow \text{NULL}$ 
for  $\mathbf{x}_i \in \mathbf{X}$  do
     $\mathbf{score} \leftarrow \mathbf{score} \cup \frac{\text{av-reach-dist}_{MinPts}(\mathbf{x}_i)}{\text{mean}_{\mathbf{x}_l \in N_{MinPts}(\mathbf{x}_i)} \text{av-reach-dist}_{MinPts}(\mathbf{x}_l)}$ 
end
// Standardization
 $\mathbf{score} \leftarrow \frac{\mathbf{score}}{\text{sum}(\mathbf{score})}$ 
return :  $\mathbf{score}$ 

```

2.2.6.2 Local correlation integral (LOCI)

The following section is highly inspired by the paper *LOCI: Fast Outlier Detection Using the Local Correlation Integral*[24]. In comparison with the local outlier factor describe above, the local correlation integral is a pure density-based approach, as the area around the samples is fixed.

Multi-granularity deviation factor (MDEF) The Multi-granularity deviation factor, MDEF, corresponds to the local density variations of the feature space. This factor is defined in an area around the sample, in this section we will use a distance, noted r , to delimit it. This area is called the r -neighborhood. A sample is considered as an outlier if its MDEF is slightly different than the average MDEF from all the points in its r -neighborhood. The Multi-granularity deviation factor is useful to detect both isolated outliers and clusters.

The MDEF of a point $\mathbf{x}_i$, with a radius r , corresponds to the relative deviation of its local neighborhood density from the average local neighborhood density of all the samples in its r -neighborhood. A point with a higher MDEF is more likely to be an outlier.

Definition 2.9 (Counting neighborhood). The counting neighborhood of a sample $\mathbf{x}_i$ with a radius r , $M(\mathbf{x}_i, r)$, corresponds to the number of objects in $\mathbf{x}_i$'s r -neighborhood.

Definition 2.10 (Average counting neighborhood). The average counting neighborhood of a sample $\mathbf{x}_i$ with a radius r , and a proportion parameter α , $AM(\mathbf{x}_i, r, \alpha)$, corresponds to the average of the counting neighborhood of the sample in the r -neighborhood of $\mathbf{x}_i$:

$$AM(\mathbf{x}_i, r, \alpha) = \text{mean}_{\mathbf{x}_l \in r\text{-neighborhood}(\mathbf{x}_i)} M(\mathbf{x}_l, \alpha * r)$$

Note that the point $\mathbf{x}_i$ is contained in its own neighborhood that implies that no one can be empty. The figure 2.6a shows a simple dataset and the corresponding counting neighborhood for some points. $\alpha * r$ is called the sampling neighborhood. It is a constant related to r whatever is size is, and is a critical parameter of the algorithm.

Definition 2.11 (MDEF). For any $\mathbf{x}_i, r$ and α the Multi-granularity deviation factor is defined as:

$$\begin{aligned} \text{MDEF}(\mathbf{x}_i, r, \alpha) &= \frac{AM(\mathbf{x}_i, r, \alpha) - M(\mathbf{x}_i, \alpha * r)}{AM(\mathbf{x}_i, r, \alpha)} \\ &= 1 - \frac{M(\mathbf{x}_i, \alpha * r)}{AM(\mathbf{x}_i, r, \alpha)} \end{aligned} \quad (2.11)$$

In figure 2.6b, MDEF is shown as the relative difference between the two curves. This illustration is a good way of seeing the importance of the value of $\alpha * r$ because:

- with a small value of $\alpha * r$, M and AM have the same value which is equal to one as there is only one point in the neighborhood;

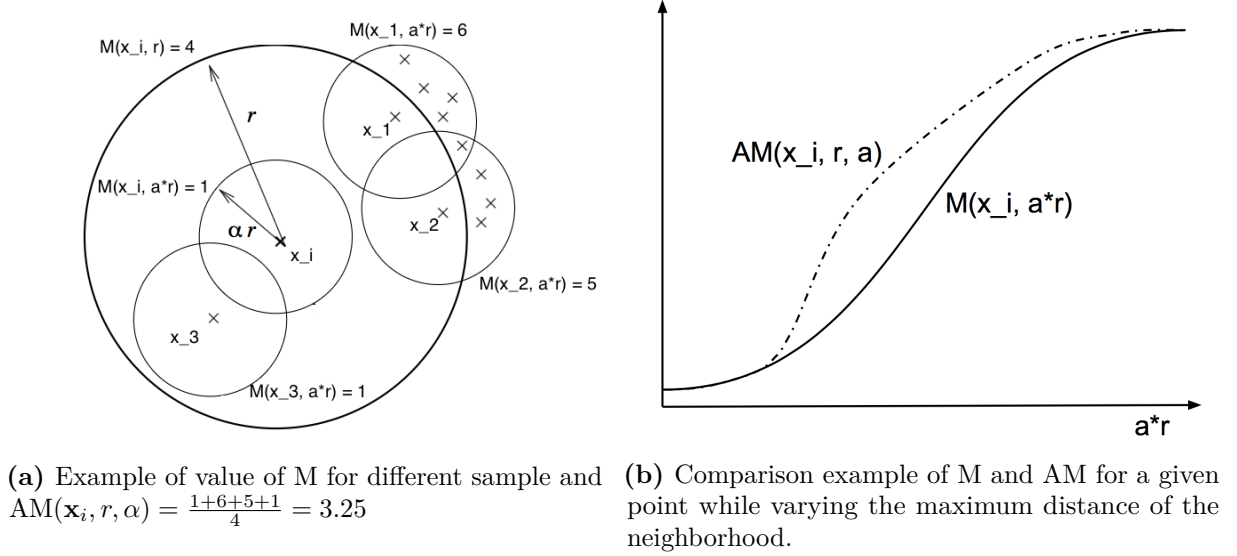


Figure 2.6: LOCI representation

- with a medium value of $\alpha * r$, M and AM could have really different values;
- with a large value of $\alpha * r$, M and AM have the same value which becomes constant and corresponds to the situation when all the points are in the neighborhood of all the points;

Normalized standard deviation This outlier detection algorithm relies on the standard deviation of the αr -neighbor count over the sampling neighborhood of $\mathbf{x}_i$.

Definition 2.12 (σ_{MDEF}). For any $\mathbf{x}_i, r$ and α the normalized standard deviation is defined as:

$$\sigma_{MDEF}(\mathbf{x}_i, r, \alpha) = \frac{\sigma_{AM}(\mathbf{x}_i, r, \alpha)}{AM(\mathbf{x}_i, r, \alpha)}$$

$$\sigma_{AM}(\mathbf{x}_i, r, \alpha) := sd(AM(\mathbf{x}_l, r, \alpha)), \forall \mathbf{x}_l \in r\text{-neighborhood}(\mathbf{x}_i)$$

α is contained in the interval $]0; 1[$ to deal with some singularities in the sample space. In general its value is fixed to 0.5 for a fast computation.

LOCI outlier detection To use this algorithm, the neighborhood and the way of spotting outliers must be defined.

- size of the neighborhood: In figure 2.6b it can be seen that taking a too large or a too small neighborhood can't lead to satisfying results, in practice several values of distances (e.g. having $r_{min} = 20\%$ to $r_{max} = 80\%$ of the dataset as its neighbors) will be tested;
- there exists two ways to check if a point is an outlier:
 - a point is flagged as an outlier if $\forall r \in [r_{min}, r_{max}]$ $MDEF$ is sufficiently large: $MDEF(\mathbf{x}_i, r, \alpha) \geq k * \sigma_{MDEF}(\mathbf{x}_i, r, \alpha)$. In general k is fixed to 3 under the assumption of a normal distribution. The advantage of this technique is the fact that the threshold is clearly defined for a normal distribution;
 - but it is also possible, and desirable in our case (to avoid any normal assumption and gain information), to use a ranking instead of a hard limit. In that case, the mean value of $\frac{MDEF(\mathbf{x}_i, r, \alpha)}{\sigma_{MDEF}(\mathbf{x}_i, r, \alpha)}$ while varying r between $[r_{min}, r_{max}]$ is used as the outlier score;

Algorithm 7: LOCI ranking algorithm

```

input   :  $\mathbf{X}$  the set of input samples
            $\alpha$  portion of the distance of the neighborhood,  $< 1$ 
            $\mathbf{r}$  vector of radius from  $r_{min}$  to  $r_{max}$ 
output : score a vector of score for each sample
score  $\leftarrow$  NULL for  $\mathbf{x}_i \in \mathbf{X}$  do
    score  $\leftarrow$  score  $\cup$  mean(for  $r \in \mathbf{r}$  do  $\frac{MDEF(\mathbf{x}_i, r, \alpha)}{\sigma_{MDEF}(\mathbf{x}_i, r, \alpha)}$ ) // Standardization
    score  $\leftarrow$   $\frac{\mathbf{score}}{\text{sum}(\mathbf{score})}$ 
end
return : score

```

LOCI algorithm The worst-case complexity of this algorithm is $O(N(\text{time_of_}r_{max_range_search} + M_{ub}^2))$, where $M_{ub} = \max(M(\mathbf{x}_i, r_{max}) | \mathbf{x}_i \in \mathbf{X})$.

As this complexity is quite heavy, an optimization can be done by estimating M and AM . It is called ALOCI.

2.2.7 Ensemble methods

Most of the material presented in this section is coming from the work found in [2] and [32].

No single method can detect all types of outliers : every algorithm relies on a model based on some assumptions to qualify outliers. This is one of the reasons of why ensemble methods are so useful in outlier detection. They are techniques which, based on the predictions from other methods (also called detectors), compose a new, ideally better, prediction. The notion of what a better prediction is depends of the expectations regarding the results, the ensembles can be used in two optics:

- increase the diversity of the results by integrating the different results discovered by the base detector to find more outliers;
- increase the confidence in the results and the robustness by favoring the points detected by multiple detectors.

2.2.7.1 Diversity of usage

Ensemble methods are a corner stone of a lot of algorithms requiring composed results such as:

- **bagging algorithms:**
 - **Feature bagging** allowing the base detectors to be applied in restricted subspaces and therefore reducing the impact of the curse of dimensionality;
 - **Sample bagging** which reduces the amount of examined samples and can speed up the detection;
- **Algorithm sensitive to the initialization** such as clustering algorithms, which requires several random runs in order to diversify their results;
- **Models learned with different parameters** is done in many base detectors, like for example in LOF and LOCI to change the neighborhood granularity;
- **Different model ensemble** using different algorithms to build more robust or diversified results.

All these possible types of ensembles are not used in the same way, and do not have the same objectives

2.2.7.2 Base detector selection

The choice of base detectors used for the composition of the ensemble can be the result of a manual or automated selection.

We won't go more into details on which techniques are used by several algorithms requiring a score composition and will focus on the techniques that can be applied as a final step for an outlier detection using multiple models to try to increase the quality of our results.

The first, obvious option to select the base detector incorporated in the ensemble is to do it manually after comparing the results of the different techniques. This allows to :

- reduce the computational time by removing redundant techniques;
- exclude techniques which always underperform.

Algorithm 8: Greedy model selection

```

input :  $L$  the outlier detectors list
         $t$  the number of points taken into account by base detectors
         $n$  the number of samples
output :  $E$  a subset of base detectors returned for the ensemble
//  $t_l$  correspond to the  $t$  samples with highest scores, for base detector  $l$ 
 $E \leftarrow \emptyset$ 
 $T \leftarrow \cup_{l \in L} t_l$ 
 $\mathbf{v} \leftarrow$  target vector //  $v_i = 1$  if sample  $i \in T$ , 0 otherwise
Sort  $L$  by weighted Pearson correlation to  $\mathbf{v}$ 
// Outliers weight:  $\frac{1}{2*|T|}$ , inliers weight:  $\frac{1}{2*(n-|T|)}$ 
// the getFirst function removes and returns the first element of a list
 $E \leftarrow E \cup \text{getFirst}(L)$ 
 $\mathbf{p} \leftarrow$  prediction vector of  $E$ 
Sort  $L$  by weighted Pearson correlation to  $\mathbf{p}$  (decreasing order)
while  $\text{length}(L) \neq 0$  do
     $\text{score}_1 = \text{getFirst}(L)$ 
    if  $\text{weightedPearson}(E \cup \text{score}_1, \mathbf{v}) \geq \text{weightedPearson}(E, \mathbf{v})$  then
         $E \leftarrow E \cup \text{score}_1$ 
         $\mathbf{p} \leftarrow$  prediction vector of  $E$ 
        Sort  $L$  by weighted Pearson correlation to  $\mathbf{p}$  (decreasing order)
    end
end
return :  $E$ 

```

Another solution is to use a greedy approach with weighted similarity measures to automatically detect the useful base detectors. The algorithm of the greedy model selection can be found in algorithm 8. This approach uses a base vector, containing the t highest score samples of each method, to compute the similarity measure, based on weighted Pearson. Then, the base detector closest to the base vector is set as the first ensemble item. The next step is to find the basic detector with the lowest correlation with the current ensemble and test if this detector must be included based on whether it increases the correlation with the base vector or not.

2.2.7.3 Score harmonization

When working with different types of base detector, the first concern in order to build the ensemble's result is the normalization of the scores as the scores returned by the different algorithms usually do not have the same scale.

It exists three main methods to make the score comparable:

1. use a sigmoid function coupled with mixture modeling on each score vector;
2. scale each score vector using the standard deviation;

3. reason on the score distributions.

This step is important to avoid a bias in the final decision. It exists types of ensembles that only use ranking instead of score to avoid this normalization, unfortunately this approach leads to a loss of information as we lose the notion of confidence that is usually expressed by the distance between the scores.

2.2.7.4 Scores combination

The last concern will be the score combination. The most common approaches are averaging or maximizing the vectors. Both have their advantages and drawbacks:

Averaging aims at reducing the variance in the results, and thus, increases the robustness. The final score of a sample $\mathbf{x}_i$ can be computed as $\text{mean}(\mathbf{x}_i) = \frac{\sum_{l=1}^{|L|} \text{scoring}_l(\mathbf{x}_i)}{|L|}$, with $\text{scoring}_l(\mathbf{x}_i)$ the score given to the sample $\mathbf{x}_i$ by the detector l . The main drawback of averaging is the fact that local anomalies can be hidden.

Maximizing aims at increasing the diversity by flagging all base detector anomalies. Unfortunately, its effect is much harder to describe as it increases the variance. The final score of a sample $\mathbf{x}_i$ can be computed as $\text{max}(\mathbf{x}_i) = \max_{l \in L} \text{scoring}_l(\mathbf{x}_i)$, with $\text{scoring}_l(\mathbf{x}_i)$ the score given to the sample $\mathbf{x}_i$ by the detector l . Maximizing approach is more often used when working with ranking to limit the variance increase.

Working with weighted similarity measures, like weighted Pearson, to compare base detectors output is highly recommended as inlier and outlier scores should not have the same impact on the final combination.

2.3 Curse of Dimensionality

Another important point to take into consideration when evaluating which technique might be used or not in REED is the dimensionality issue.

The first intuition, when dealing with a problem where we don't have any specific information about the data distribution, is to use proximity based methods. However, that family of methods is known to be extremely sensitive to the curse of dimensionality and more precisely to the distance concentration in high dimensionality. As seen in the *Machine Learning : regression, dimensionality reduction and data visualization* course:

$$\lim_{d \rightarrow \infty} \frac{\sqrt{\text{var}(\|\mathbf{x}\|_2)}}{E(\|\mathbf{x}\|_2)} = 0 \rightarrow \frac{d_{\max} - d_{\min}}{d_{\min}} \rightarrow_p 0 \quad (2.12)$$

That implies that all the pairwise distance becomes more similar in high dimension, which is a real issue for all the detectors that uses distance comparison. There exists at least two kinds of behaviors observed if this problem is met:

- The first behavior, encountered mainly in the hierarchical (and common) clustering approaches, will be that the decisions will become almost random. As all the points will have approximately the same pairwise distance with all the points, the algorithm will make random decisions that will lead to a random tree. This behavior is the worst as it will lead to really poor results due to the unbalancing problem.
- The second behavior, encounter by two phases clustering and density based approach, will result in the inability for the algorithm to detect any outliers.

The two phases of the clustering algorithm will be studied individually. The first phase will be purely random, as the initialization is random and as the moving process will be too.

The second phase will have a positive effect. As all the distance between the centroids will be almost constant, the set will have very little cut, and thus, will lead to close scores. For density based approach, the same conclusion can be taken. If the distance between all the points is almost equal, we can conclude that the density perceived by the algorithms will be almost constant. Thus, all the observations will have approximately the same score.

The second behavior is much more desirable in the case of an ensemble, as this conclusion, will not downgrade the final solution.

2.3.1 Dimensionality Reduction

Different approaches exists to reduce the dataset dimensionality, some of them being:

- **features transformation** for example, projecting the data into a lower-dimensional space;
- **feature selection** reducing the feature space by removing the features containing little to no information.
- **using feature bagging** techniques to split the analysis in multiple parts and recombine the results

Chapter 3

Implementation

This chapter will describe what implementation choices we made based on the techniques presented in the previous chapter. The code of our implementation can be consulted on <https://github.com/aure-pigno/outlier-detection/>. The git contains a "read me" providing information on the structure of the code and the dependencies, the datasets used, the stored computed raw results and the documented source code.

We will start by quickly explaining the structure of the code in section 3.1, will continue with the description of the data transformation performed on the datasets in section 3.2 and will finish with the description of the implemented methods in section 3.3.

3.1 Code structure

In order to make our code more easily (re)usable, it is structured in two packages:

outliers containing the methods used to perform the outlier detection. The methods are separated in files based on their family of method.

validation The validation package contains the methods to simulate synthetic datasets, introduce noise or outliers and all the visualization methods used in this work.

We tried, as much as possible, to avoid redeveloping code on our own and used existing R packages when possible and followed the recommendations from [29] to structure our packages. All our methods are documented and a script `make_package.R` is available to easily recompile and reload the packages after a modification.

3.2 Pre-processing

As the handled datasets have different sources, we first converted them in a unified format in order to be able to automatize the experiments. All the datasets have been saved as data frames with named rows and columns and stored in a series of files using the `save` method. These files can then be loaded as needed. For supervised datasets, the file contains a list with two objects, `x` the dataset and `y` a boolean vector indicating the outliers.

The pre-processing steps described in the following subsections are not applied on the raw saved dataset but are performed before calling the detection methods as the same pre-processing is not required for all the methods.

3.2.1 Data transformations

Categorical features Most of the techniques are not able to handle categorical data. There are different ways to replace them with a numerical representation, we chose to replace them

with a set of boolean. For example, `color` with the factors `blue` and `red` and `yellow` becomes two variable `color.blue` and `color.red` with 1 if the initial categorical variable had that color and 0 otherwise. An extra variable for the last factor, `yellow` is not necessary as the variables are mutually exclusive. `yellow` is therefore represented by the absence of both `red` and `blue`.

This way most of the initial information is preserved, and the variable is usable by methods only able to handle numerical attributes. The only information lost is the relation. As there is no way for the model to know the relation between those new variable, the fact that they are mutually exclusive is not represented anymore.

Low variance features As the features with really low to zero variance are not offering any additional information and can slightly increase the computation time, they are removed from the dataset during the pre-processing.

Exponential distributions To deal with non-normal distribution, we have considered several normalization techniques:

- nothing;
- a BoxCox transformation;
- the inverse hyperbolic sine over the entire set of features;
- the inverse hyperbolic sine on feature with a high deviation.

The BoxCox transformation being frequently used as a pre-processing step in code examples. The impact of those transformations on an exponential distribution is represented in figure 3.1.

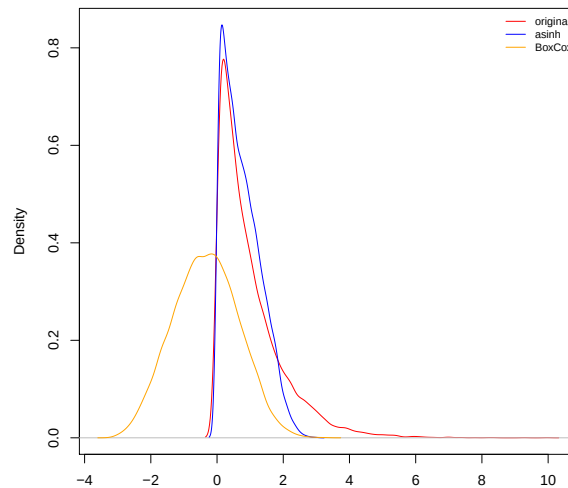


Figure 3.1: Different transformations of an exponential distribution

Rescaling the dataset is useful to prevent a variable with large numerical values to dominate the others in distance-based methods.

It is important to remember that the pre-processing is truly independent of the detection methods, and could be applied differently for each detector, as these detectors are sensitive to different types of data.

3.2.2 Dimensionality reduction

As a lot of techniques are negatively affected by a high-dimensional space, we have investigated two ways of reducing it.

PCA The first way is by first performing a PCA and applying the detectors on the data projected on the new feature space.

Features bagging The second way, already introduced in the state of the art, is feature bagging. The feature space is separated in several bags on which the detection is performed. Once done, the results are recomposed using an ensemble method. The bagging has been performed in order to prevent the dimensionality to be higher than the number of samples. The results from the different bags have been composed using an ensemble method with a mean as internal function and no weights on the different results. We experimented two ways of selecting the bags:

- **random feature bagging** a basic approach which randomly splits the feature space into different bags;
- **feature bagging based on the mutual information** bagging can lead to a huge improvement in the results thanks to the dimensionality reduction and the study of local solutions. In the case of outlier detection, introducing a bias to reduce the noise of the different bags is, in theory, beneficial compared to random bag.

We used maximization of the mutual information as bias, through a greedy algorithm, inside all the bags. Unfortunately, the computation of the mutual information matrix could be really heavy, for enormous matrices.

One of the dangers of bagging in general is that it can lose some information about the correlation of the features potentially leading to unstable results.

3.3 Methods

This section will present how we decided to implement the different detector presented in the state of the art, and which modification we performed, together with the justification for those modifications.

3.3.1 Two phases clustering

The outlier finding process of the two phases clustering, algorithm 4, takes only global information. But as discussed in 2.2.6 taking into account only local interaction is beneficial.

The separation phase in the outlier finding process (OFP) needs to be adapted. This change is applied to take into account the distance between the clusters. And thus, allow the presence of sparse regions without overestimating the proportion of outliers. figure 3.2 compares the two points of view. An example of this new rule could be: An edge must be deleted if its length is much greater than the one of the other edges of a given vertex/cluster. The new OFP algorithm can be found in algorithm 9.

As clustering algorithms are highly impacted by their clusters' initialization the entire algorithm is ran multiple times and a sequential ensemble algorithm is used to express the final result of the algorithm.

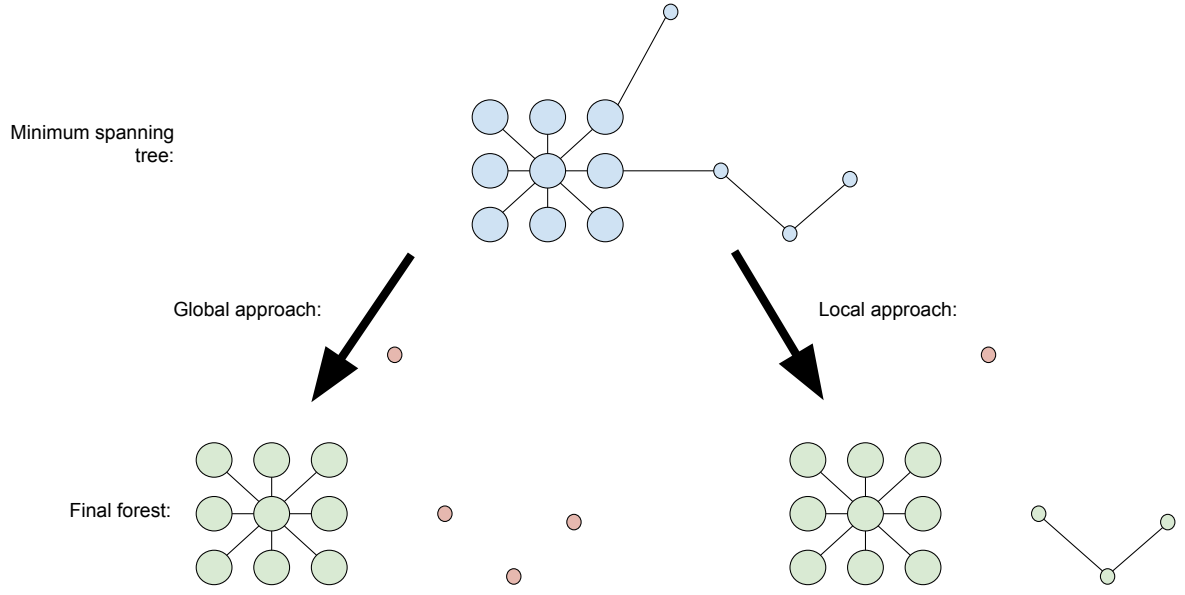


Figure 3.2: Comparison of global and local OFP

Algorithm 9: Modified OFP

```

input :  $c$  the set of centroids
         $C$  the set of clusters
         $\Theta_r$  determines the threshold above which an edge must be removed
         $\Theta_s$  determines the threshold above which the elements of a cluster must be seen as
        outliers
output : outliers the set of outliers
 $F \leftarrow \text{MST}(c)$  for  $\text{node in nodes}(F)$  do
    |  $\text{mean\_length} \leftarrow \text{mean length of the edges of node}$ 
    |  $\text{// Remove the edges higher than } \Theta_r * \text{mean\_length}$ 
end
// Merge all the clusters, with connected centroids
 $C_1..C_m \leftarrow \text{the new clusters}$ 
outliers  $\leftarrow \emptyset$ 
// Smallest subtrees regard as outliers
for  $\text{cluster} \in \{C_1..C_m\}$  do
    | if  $\text{size}(\text{cluster}) < \Theta_s$  then
    | | outliers  $\leftarrow \text{outliers} \cup \text{cluster}$ 
    | end
end
return : outliers

```

3.3.2 Hierarchical Clustering

In addition to the algorithm 5, developed in the state of the art, we decided to add an additional step to be more consistent, and a score function, instead of hard labels, to smooth the results.

- The new step is the use of an ensemble algorithm which consists of applying hierarchical clustering, on the dataset, while removing outliers at each execution, until there are no more outliers found. This modification puts the threshold $\Theta = 1$ to increase the impact on (really) isolated samples.

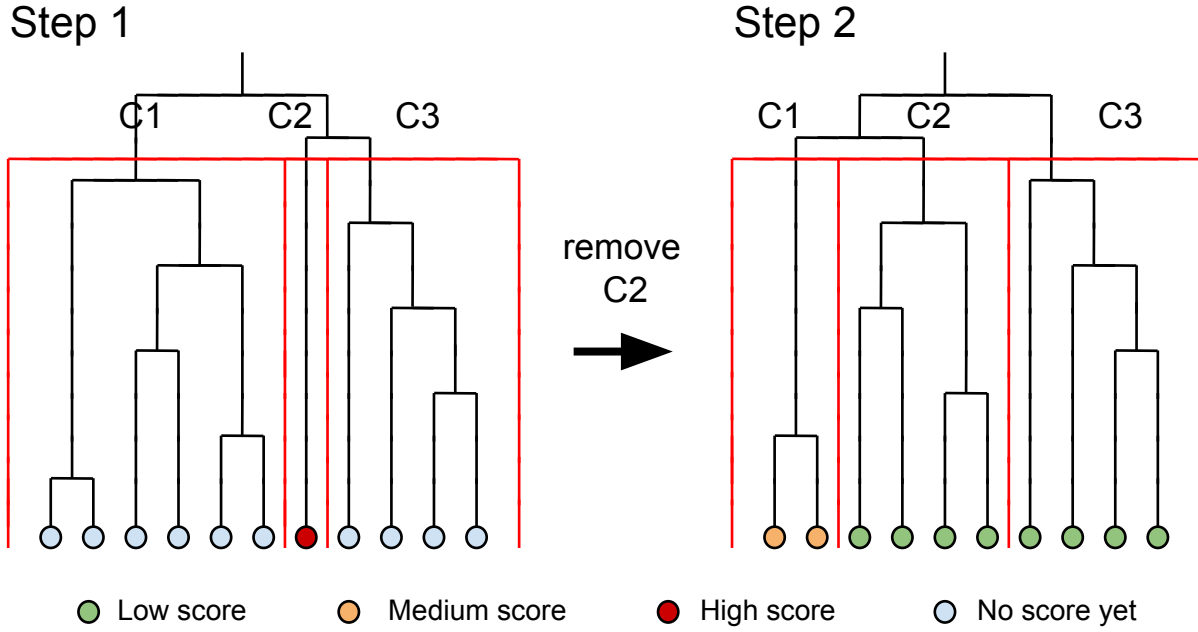


Figure 3.3: The points of cluster 2 (image 1) will be considered as outliers (score = 1)

- The scoring function is equal, for each point, to the inverse of the size of the cluster the sample belongs to. The purpose of this change is to smooth the score given to each point and to be less parametric as t has no importance (after the first step)

A graphical representation of these two new features is present on figure 3.3 while the modified algorithm can be found in algorithm 10.

Algorithm 10: hierarchical clustering

```

input :  $\mathbf{X}$  the set of  $n$  input samples
        dist a distance function
         $K$  the number of clusters
output : score a score for each  $\mathbf{x}_i \in \mathbf{X}$ 
score  $\leftarrow$  a vector of 0 of size  $n$ 
while  $\mathbf{X}$  is unchanged do
     $\mathbf{D} \leftarrow$  compute the distance matrix from  $\mathbf{X}$  by using dist
     $htree \leftarrow$  compute the hierarchy tree using  $\mathbf{D}$ 
     $hclusters \leftarrow$  Cut  $htree$  into  $K$  clusters
    for  $C \in hclusters$  do
        if  $size(C) == 1$  then
             $score[obs] \leftarrow 1, \forall obs \in C$ 
             $\mathbf{X} \leftarrow \mathbf{X} \setminus \{obs\} \forall obs \in C$ 
        end
    end
end
for  $C \in hclusters$  do
     $score[obs] \leftarrow \frac{1}{size(C)}, \forall obs \in C$ 
end
// Standardization of the output
score  $\leftarrow \frac{score}{sum(score)}$ 
return : score

```

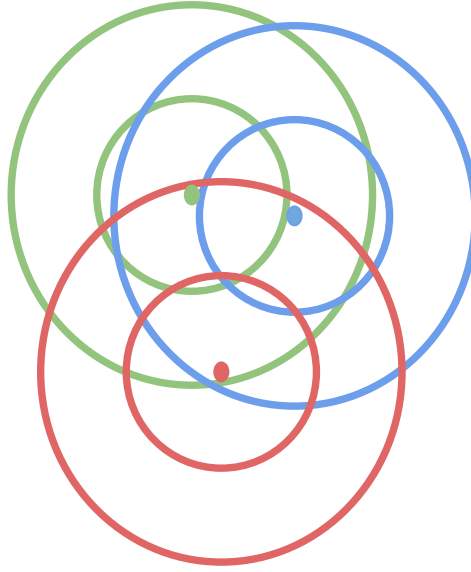


Figure 3.4: Distance concentration problem for LOCI

3.3.3 Local correlation integral

LOCI, as explained in the state of the art, suffers of the curse of dimensionality and more precisely to the distance concentration described in Section 2.3. This phenomenon implies that pairwise distances seem nearly equal for all points, which has a huge impact on LOCI because of the α which determines the proportion of distances to pass from the average counting neighborhood (AM) to the counting neighborhood (M). If the difference between the distances are not sufficiently high, all the scores will be identical. A basic two-dimensional representation is present on figure 3.4. As is shown, if the distance between the red point and the 2 others is greater than the distance between the two other points, the result of LOCI will be the same for the three points as their own M, AM and thus MDEF and θ_{MDEF} will be identical. This appears because of a not sufficiently high α . This issue is unlikely to appear in small dimensions as the norm concentration appears in high dimensionality and as the number of ($\pm$)equally distanced points is limited by the dimensionality (2 in 1D, 3 in 2D, 4 in 3D, etc). To reduce the impact of this problem, we have added another ensemble algorithm that will play on α and that finally take the mean of all the results. This approach has nevertheless some problems:

- it highly increases the computational time as the algorithm runs multiple time and as, with higher α , a higher number of points are considered as neighbors and, a higher number of neighbors are considered for all the neighbors, etc.
- Increasing α make the correlation integral less local. In really high dimensionality, if $\alpha \neq 1$ the problem will remain in very very high dimensions, and if $\alpha = 1$ the algorithm loses all its interest as it becomes a basic density comparator.

3.3.4 Local outlier factor

As LOF is known to have low performance in higher dimensionality, we included a PCA projection directly when using this method. This practice commonly used in the literature. We didn't re-implement the method as it was already available in a r package.

3.3.5 One-class SVM

We used the implementation of one class SVM present in the `caret` package without altering it. The kernel used was a Gaussian kernel.

3.3.6 PCA

We included the scoring method based on the distance to the center of a PCA as a detector using the formula described in the state of the art.

3.3.7 Ensemble

As already explained in the state of the art, different strategies can be taken in order to build the ensemble based on the expected results.

Choices of base detectors We considered two methods of selection for the base detectors included in the ensemble:

1. consider every method as a base detector;
2. uses the greedy model selection (algorithm 8, described in the state of the art) to dynamically select the base detectors.

The ensemble method we implemented takes different arguments allowing to define how the scores are composed.

Score combination All the scores returned by the base detector are normalized; the sum of all the scores returned by a detector is equal to one. This normalization has been done in order to easily give the same default weight to every detection method when performing score combination in the ensemble.

As explained in the state of the art, there exists different techniques to combine the base detector's scores. All the experiments are done with the three most famous ones which consist of computing the maximum score, the average score or the median score, for each sample over all the base detectors.

Weight for base detectors The state of the art talks a lot about the phenomenon of diversification, but we also wanted to take the opposite approach and look at score concentration. This approach consists of increasing the weight of methods that lead to common results, with, of course, a maximum and minimum weight for each detector to avoid the loss of one of them. These weights are computed with a fix point algorithm that computes the detector weight from the previous ensemble (computed initially with a constant vector of weight), and the detector score vector.

Two approaches are tested here. The first one is looking at the number of **common samples** in the top elements. In this work, this approach will be called **ensemble-c**.

The second approach is considering the **distance between the samples** contained in the top elements of the ensemble with their position in the base detectors. In this work, this approach will be called **ensemble-d**.

Of course, the two computations are done only on the sample with the highest scores as the inliers should only have a limited impact on the final result.

The fix point algorithm can be found in algorithm 11, while the two comparison algorithms can be found in algorithms 12 and 13.

Algorithm 11: Fix point

```

input : base_detectors list that contains the score vector for the  $B$  base detectors
        combi the combination function
        comp the comparison function
        min_weight the minimum authorized weight
        max_weight the maximum authorized weight
        prop proportion of elements taken into account for the fix point
output : ensemble_score vector of score after the application of the ensemble
base_weight  $\leftarrow$  a vector of  $\frac{1}{B}$  of size  $B$ 
ensemble_score  $\leftarrow$  combi(base_weight * base_detectors)
repeat
  for  $b$  from 1 to  $B$  do
    | base_weight[ $b$ ]  $\leftarrow$  comp(base_detectors $b$ , ensemble_score, prop)
  end
  // normalization:
  base_weight  $\leftarrow$   $\frac{\text{base\_weight}}{\text{sum}(\text{base\_weight})}$ 
  for  $b$  from 1 to  $B$  do
    | base_weight[ $b$ ]  $\leftarrow$   $\max(\text{min\_weight}, \min(\text{max\_weight}, \text{base\_weight}_b))$ 
  end
  base_weight  $\leftarrow$   $\frac{\text{base\_weight}}{\text{sum}(\text{base\_weight})}$ 
  ensemble_score  $\leftarrow$  combi(base_weights * base_detectors)
until no change in ensemble_score between two consecutive iterations;
return : ensemble_score

```

Algorithm 12: ensemble-c

```

input : base_score score vector of size  $n$  for the base detector
        ensemble_score score vector of size  $n$  for the previous ensemble
        prop proportion of elements taken into account for the fix point
output : weight the weight for this base detector
 $t \leftarrow \text{as.integer}(n * \text{prop})$ 
sorted_base  $\leftarrow$  sort(base_score)
sorted_ensemble  $\leftarrow$  sort(ensemble_score)
weight  $\leftarrow$  0
for  $i$  from 1 to  $t$  do
  | weight  $\leftarrow$  weight + (sorted_ensemble[ $i$ ]  $\in$  sorted_base[1 :  $t$ ])
end
return : weight

```

Algorithm 13: ensemble-d

```

input : base_score score vector of size  $n$  for the base detector
        ensemble_score score vector of size  $n$  for the previous ensemble
        prop proportion of elements taken into account for the fix point
output : weight the weight for this base detector
 $t \leftarrow \text{as.integer}(n * \text{prop})$ 
sorted_base  $\leftarrow$  sort(base_score)
sorted_ensemble  $\leftarrow$  sort(ensemble_score)
weight  $\leftarrow$  0
for  $i$  from 1 to  $t$  do
  | index_base  $\leftarrow$  index(sorted_base == sorted_ensemble[ $i$ ])
  | weight  $\leftarrow$  diff +  $\text{abs}(i - \text{index\_base})$ 
end
weight  $\leftarrow$  (weight + 1)-1 // +1 is done to avoid zero division
return : weight

```

Chapter 4

Evaluation protocol

In this chapter, we will establish what needs to be assessed in our experiments and how our validation protocol will be performed. As the aim of this work is to provide a method that can be integrated in a real live system, the computational performances of the algorithms as well as the quality of their results need to be assessed in order to have a good vision of which techniques are more efficient than others. Our evaluation protocol therefore has to consider the following aspects:

- the computational time;
- the quality of the outlier detection, which should, ideally:
 - maximize the true positives
 - minimize the false negatives
 - maximize the stability of the results

The best solution being a trade off between those different aspects.

We will start by a survey of the evaluation techniques encountered in the literature in section 4.1, follow with our evaluation protocol in section 4.2 and finish with a more detailed explanation of the statistical tests and visual representation we will use in sections 4.3 and 4.4

4.1 Different evaluation protocols seen in the literature

4.1.1 Datasets

As the outlier detection is typically an unsupervised problem, there is no handy labeled public dataset available for this task. Most of the evaluation methods encountered in the papers were based on two different kinds of datasets:

- Public datasets ¹
 - with the smaller class used as outliers [22] [32] [11] [20] [19] [3] [31] [26] [4]
 - with artificially added outliers and noise randomly [13]
- Synthetic datasets: different patterns with varying [11][24][13] [22] [20] [19] [26]
 - level of noise
 - number of outliers

¹frequent example: UCI ML Repository

4.1.2 Validation method

Most of the outlier detection algorithms are returning scores and not a binary decision. It is therefore not possible to assess the performances of an algorithm based on the accuracy of the prediction as the accuracy²(or any other measure based on binary decisions) will be dependent off a threshold set for the decision.

To tackle this issue, most of the papers are using the area under curve to compare the performances of the detection methods:

- ROC curve with comparison of the AUC [32] [20] [19] [4]
- precision at k (How many of the top k ranked data objects are actual outliers) [32]
- using the F-measure: ³: $F = \frac{2 * R * P}{R + P}$ [13]
- compare the performance of a classifier after removing outliers [13]
- compare results using (PCA +) LOF as baseline[24] [11]

Most of those methods require to know the labels, which is why most of the papers are using samples from a minority class as outliers or generated synthetic datasets/outliers.

4.2 Validation protocol

We will follow the examples from the literature and perform an evaluation of the performances of the different detectors based on both public files considering the minority class as outliers and on simulated datasets in order to be able to perform a supervised evaluation. We will also try to assess the performances in an unsupervised fashion as described in section 3.4.

4.2.1 Supervised

As visible in the subsection 4.1.1, the AUC is a popular measure to evaluate the performances of the different methods. According to [7], the AUC, appears to be one of the best ways to evaluate a classifier's performance, or, in our case, of an outlier detector, when no threshold has been determined yet.

A ROC (Receiver Operating Characteristic) curve provides a way to visualize a classifier's performance without having to define a decision threshold. It is a two-dimensional representation based on the sensitivity (the true positive rate) and the specificity (the true negative rate) of a classifier as the detection threshold varies.

The AUC (Area Under Curve) allows to express the overall accuracy of a classification algorithm without requiring to select a specific threshold. It is the measure of the area covered under the ROC curve and varies from 0 to 1. (1 being a perfect prediction)

The ROC curve and AUC have some interesting properties:

- as the sensitivity and specificity are success and failure rates. They are not impacted by unbalanced classes.
- in ROC space, the diagonal line $x = y$ corresponds to the performances of a random classifier[14] and provides an AUC of 50%

²For outlier detection, the accuracy is anyway not a good measure as the outlier / normal class are typically highly unbalanced

³If a threshold is known or if the return is binary. R = recall, P = precision

Using this measure seems like a perfect choice in this case as it avoids mixing up the evaluation of the detectors' performances and the quality of a decision threshold. [14] however warns of the danger to solely rely on the result of a ROC curve / AUC to compare classifiers. It cannot be concluded that a classifier is superior to another only by looking to one AUC value as it doesn't provide any information on the variance of the performances. To tackle this issue, the difference in performances must be tested on several datasets or using cross validation and evaluated using a statistical test to determine if the variations in performances are statistically significant.

Another reason why only looking at the AUC performances is dangerous in this case is that, for outlier detection, we are typically only interested in the quality of the outlier detection at the beginning of the curve. The figure 4.1 displays two different roc curves that illustrates this perfectly. The first image has an AUC of 69.7% which is, if we look at the AUC only, less good than the second one but looking at the curve, it returns a good number of outliers in the first results while the second failed to detect any outliers in the higher scores.

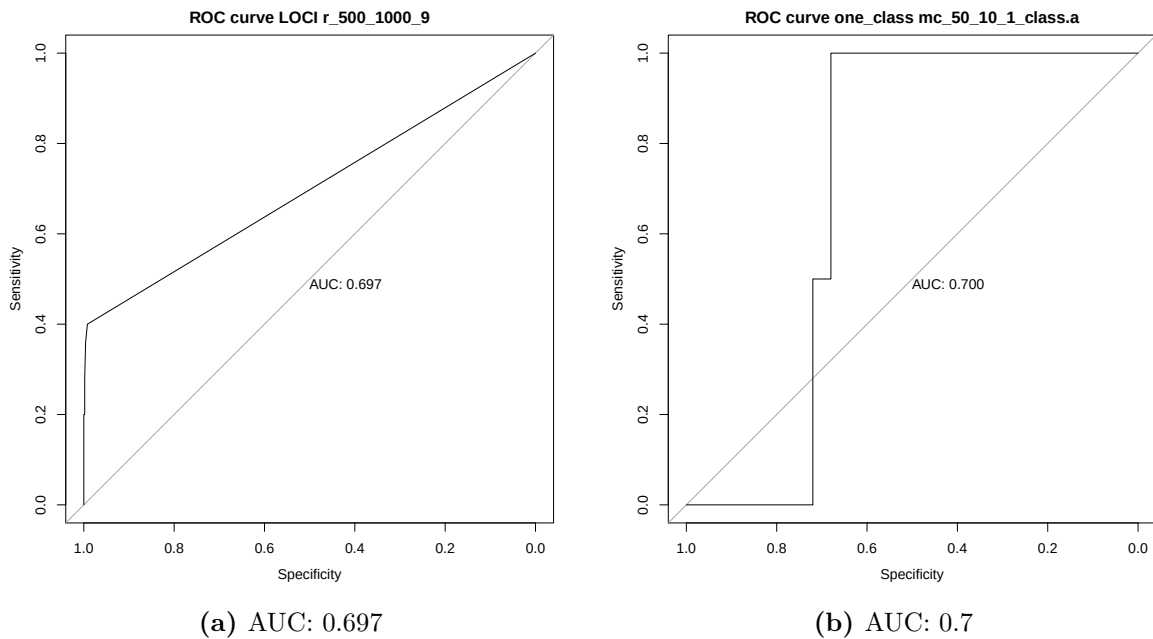


Figure 4.1: Two roc curves illustrating the danger of using only the AUC as performance measure in outlier detection

For this reason, the quality of the different methods will be compared both by looking at statistical tests, performed on detection over several simulated and public datasets, and by looking at the precision at k on 3 different thresholds.

The simulated datasets contain 5% of outliers and the tested thresholds will be 1% , 5% and 10%.

For each evaluated algorithm, we will report the average AUC on a variety of datasets, a statistical test(described in the section 4.3) performed on those AUC to establish which algorithms significantly improve the results and visual representations of those AUC under the form of box plot to better see the variance in the results.

For the public datasets, we will also report a representation of the outlier detection on a PCA for visualization purpose.

4.2.2 Unsupervised

As already mentioned in section 4.1.2, there is no ground truth for outlier detection on real datasets, which is the reason why most of the articles are assessing the performances of the outlier detection based on experiments on real public datasets, considering the minority class, with a potential down sampling, as outliers.

This approach is based on the assumption that the behavior of the two classes is significantly different and that the classes can be separated. In real datasets, it is often not true and classes really frequently overlap, causing samples in the overlapping region to be labeled as outliers while, actually, their behavior is corresponding to the behavior of the "normal" points. If this is the case, it is impossible for the detection algorithm to detect the samples located in the overlapping region.

Additionally, when using public datasets, there is no knowing on whether the class labeled as "normal" contains real outliers or not. If it does, the accurate detection of those points by the detection algorithm would be considered as a mistake and penalized.

In [32], the authors deplore the fact that there is, at their knowledge, currently no efficient evaluation protocol for outlier detection on real datasets.

Working with synthetic datasets for the evaluation provides the necessary ground truth but introduces other biases: the generative process for the outliers corresponds to the author's vision of what an outlier might be. It thus restricts the tested outliers to anticipated cases potentially leading to over-optimistic results for some techniques (if this vision is influenced by the knowledge of the detection algorithms) or, quite the opposite, really low performances.

Using both public (with minority class as outliers) or synthetic datasets also introduce another problem: the datasets handled do not exactly correspond to the ones encountered by the application.

4.2.2.1 Classification performance

In the literature, two main motivations are often cited to justify the need of outlier or anomaly detection. The first one is to learn new data behavior, in this case the problem is considered as novelty detection and the goal is to improve the analysis by understanding and integrating the behavior of those diverging points; the second one is to remove points for which the abnormal behavior might negatively impact the analysis.

This second vision is the one that corresponds to our situation as the goal of the outlier detection in REED is to remove the abnormal samples, considered as corrupted, in order to improve the software's automated analysis.

As the outliers are affecting the classification (or regression) performances, it seems reasonable to expect that removing them has as a positive impact on their performances.

We therefore decided to include a black-box approach in our experimental process by testing the classification performances of a selection of datasets before and after removing the outliers from the dataset. This approach does not assess the quality of the outlier detection directly but tries to determine it indirectly by looking at its impact on a classification task.

This technique is, however, not completely reliable but only potentially provides an intuition on whether the detection is effective or not as it is based on the assumption that the outliers have a negative impact on the classification performance which is not always the case. Figure 4.2 illustrates this by showing a really simple two-dimensional toy example featuring two classes. The samples from each class are represented in a different color, green and blue and their outliers (1 percentile) are represented respectively in red and orange. The figure displays 4 different images of the two clusters with increasing cluster separation. On those pictures, you can clearly see that the "outliers" on the external borders of their cluster won't have a negative impact on the classification for most of the classification techniques. Only the points located around the overlapping region could potentially harm the classification performance by altering the classification boundaries.

Additionally, a global outlier analysis, not taking the classes in consideration might not be efficient to detect points located around the decision boundary as, if the clusters are overlapping, those points might look absolutely normal at the dataset level.

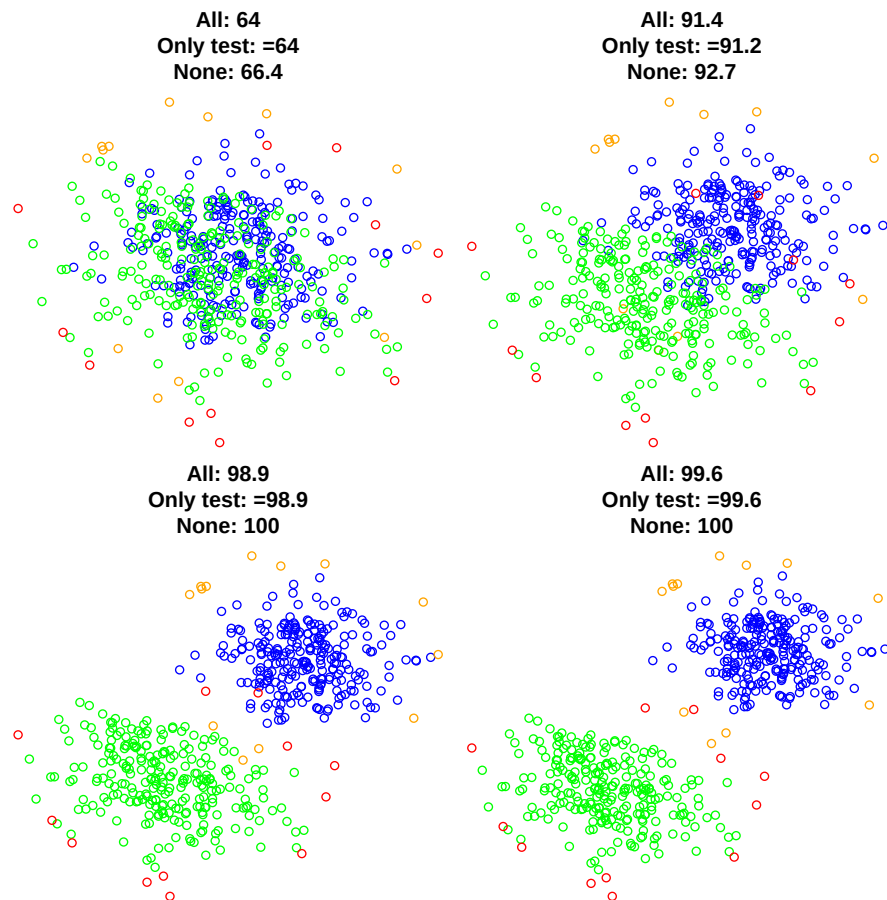


Figure 4.2: 2 dimensional Illustration: 2 clusters, each of them having their outliers and the impact of the outlier removal on the classification performances. All is the results from the classifier on the full dataset, only test removed the outliers from the training set but kept them in the test set, none does not contain any outliers. The two clusters are represented in green and blue, with their respective outliers in red and orange.

Removing outliers from the test set or training set only? The toy example from figure 4.2 displays three classification results performed on two clusters of points. those three results corresponds to an LDA classifier trained with the following setup

All The outliers are present in both training and testing set;

Only tests The outliers have been removed from the training set only;

None The outliers have been removed from the training and testing set.

The results on this toy dataset, done using a repeated 10-fold cross-validation, consistently shows (see table 4.1) that removing the outliers from the training set only is providing small to no improvement while removing the outliers from both sets consistently (although only slightly) improves the classification results. This intuitively seems logical as there is no reason for the classifier to be more efficient at classifying those points if it didn't encounter any of them in the training set.

Outlier removal on miss-categorized points Miss-classified samples can potentially have a higher impact on classification performances, but as visible on table 4.1, the impact is not really sensible when using LDA as a classifier. SVM is more impacted due to the way it computes the boundaries.

perturbation	classifier	Friedman p.value	Nemenyi CD	Full	Only test	None
external	lda	0.000	0.741	1.55	1.45	3.00
external	svm	0.000	0.741	1.35	1.65	3.00
shambles	lda	0.195	0.741	1.85	1.85	2.30
shambles	svm	0.003	0.741	1.90	1.55	2.55
extreme	lda	0.000	0.741	1.40	1.60	3.00
extreme	svm	0.000	0.741	1.35	1.65	3.00

Table 4.1: Statistical tests on 20 repeated 10-fold classifications on two, two dimensional classes with a small overlap (corresponding to the second picture in figure 4.2). in the perturbation types, "external" considers 1 percentile at the extremes of the two clusters as outliers, shambles switched the classes for 5% of the points and "extreme" multiplies 5% of the points by a value between 5 and 10 standard deviations.

As removing the outliers from the test set only does not seem to produce a significant improvement and as, in the set up of REED, the outliers are removed from the dataset before performing any classification task. We will only perform comparison of the classification performances when removing all the detected outliers. A statistical test will be performed to assess if any significant improvement has been archived. The resampling methods comparison performed in [23] tend to conclude that k-fold is more stable for small datasets with large set of samples; we will therefore compare the average performance per dataset of a 10-fold cross validation when comparing the classification results.

4.2.3 Benchmark

Even though the results computed by REED are not returned to the customer in real time, and thus, the computational time - as long as it remains reasonable - is not a primary concern for DNAlytics, having an implementation that is computationally efficient is important.

Therefore, in our evaluation protocol, each method will be called via a wrapper performing benchmarking. A comparison of the time required by all the tested methods on different samples/features size will be presented in table 5.22 in the experiment chapter 5.

4.3 Statistical test

In order to establish if there is a real difference in the performance of the different implemented algorithm, we need to use statistical tests as we cannot solely rely on a comparison of average values. The variance in performance can be really high as the algorithms' performances are dependent of the datasets.

It is therefore really important to perform pairwise tests as algorithms need to be compared on the same data instances.

Additionally, [12] warns against using t-tests when comparing multiple classifiers. Using the same statistical test to assess multiple hypotheses increase the probability of detecting a significant result by chance and can therefore lead to erroneous conclusions.

As illustrated in [6], if you perform 20 comparisons with a 95% confidence interval, you have a probability of $0.95^{20} = 0.36$ that none of them will be significant. Which means that there is 64% chance of getting at least one significant result.

Methods exist, such as the Bonferroni correction (equation 4.1) to adjust the value of α to take this probability in consideration but if the different tests are highly correlated, this correction can miss real differences.

$$\alpha_{new} = \frac{\alpha}{m} \quad (4.1)$$

It is therefore better to use tests specifically designed to handle multiple comparison such as the repeated measure ANOVA or the Friedman test.

The Friedman test is a non-parametric statistical test that can be used to evaluate the performances of multiple algorithms over a set of datasets, unlike ANOVA which makes assumptions about the distributions, the Friedman test only uses the ranks of the different algorithms to perform the comparison.

This reduces the bias introduced by extreme values when the tested variables do not follow a normal distribution.

In the Friedman test, the hypotheses are the following :

H₀ The (unknown) distributions are the same across repeated measures

H_a The distributions across repeated measures are different

$$R_j = \frac{1}{N} \sum_i r_j^i \quad (4.2)$$

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 \frac{k(k+1)^2}{4} \right] \quad (4.3)$$

In the equation 4.2, the rank r_i^j is the rank of algorithm j for the dataset i. The test then compares the average rank for each algorithm R_j which should be equal if the null hypothesis holds.

The Friedman statistic is distributed according to χ_F^2 with k-1 degrees of freedom, when N and k are big enough (N > 10, k > 5).

[12] states that, in theory, the Friedman test is less powerful than ANOVA when the ANOVA assumption holds but it's not especially the case when they are not.

As the Friedman test only gives information (see example in listing 4.3) on whether the null hypothesis holds or not, a post-hoc test needs to be performed if the test is significant in order to determine which methods are significantly different.

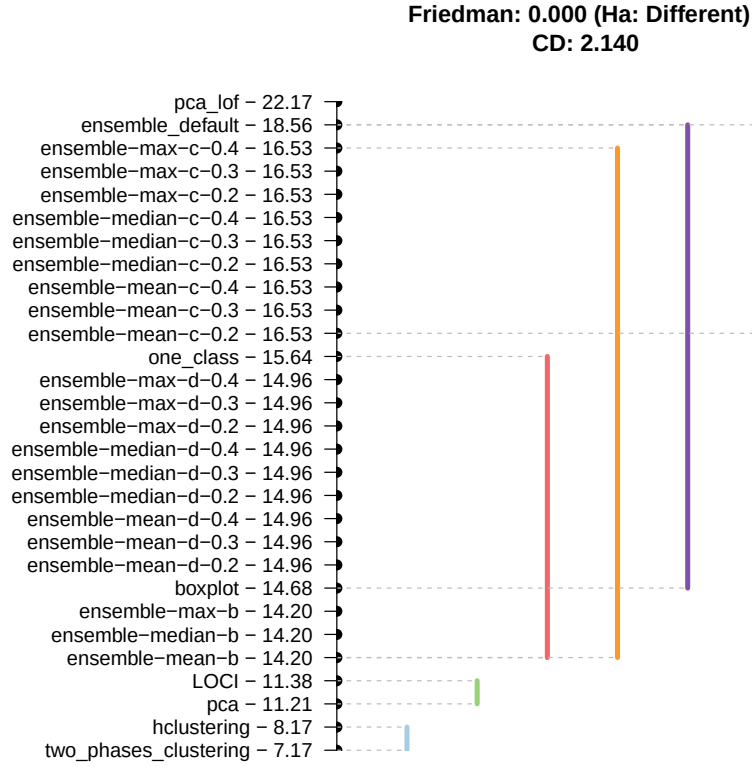


Figure 4.3: Representation of the Nemenyi results. sorted by average rank, the lines are connecting the results which are not significantly different

```
> friedman.test(auc_matrix)

Friedman rank sum test

data: auc_matrix
Friedman chi-squared = 1890.4, df = 28, p-value < 2.2e-16
```

Determining which method is significantly different from the others can be done using the Nemenyi [12] post-hoc test using the critical value in equation 4.4 to establish if the difference of rank between a pair of classifier is significant (equation 4.5).

$$CD = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}} \quad (4.4)$$

$$|R_j - R_i| \geq CD \quad (4.5)$$

The critical value, q_{α} being based on the studentized range statistic divided by $\sqrt{2}$.

In figure 4.3, a line is joining the detectors which are considered as being similar. The confidence interval in the ranks is considered to determine which detectors are similar or not. As the range goes above and under the average rank, the detectors at the extremities might not always be in the confidence interval of each other while they both are in the confidence interval of the detectors in the middle of the lines. This is why there might be duplicated lines covering the similar sets of detectors. Note that the ranks are inverted, the best detector having the higher rank which seems counterintuitive but does not impact the evaluation as inverted ranks would have identical critical distance.

We will use the Friedman with Nemenyi post-hoc test when comparing multiple algorithms or configurations and will use a t-test when comparing 2 sets of results.

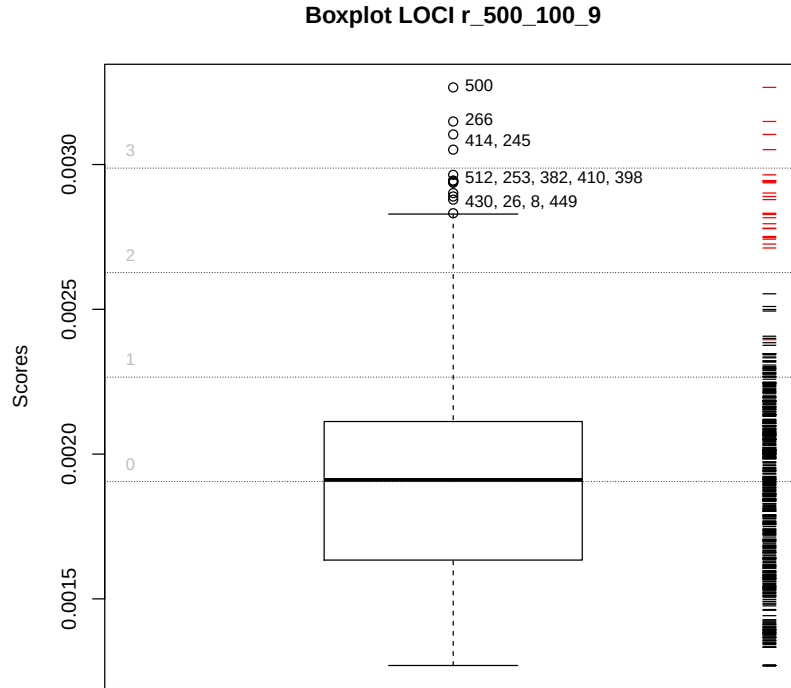


Figure 4.4: Example of a boxplot visualisation of the scores. The indexes of the samples are indicated for the samples outside of the boxplot. The real outliers are visible in red on the right side of the plot

4.4 Visualization

4.4.1 Boxplot of the scores

As stated in [1], extreme value analysis plays an important role in most outlier detections algorithm as a final step. This is because most outlier modeling algorithms quantify the deviation of the data points from the normal patterns in the form of a numerical score. Extreme value analysis is usually required as a final step on these modeled deviations, since they are now represented as univariate values in which extreme values correspond to outliers. Therefore, even though the original data is not in a form where extreme-value analysis is directly helpful, it remains an integral part of the outlier detection process.

Displaying the scores in a boxplot is therefore a sensible way to visualize the results from detection algorithms as they provide an intuition of how confident the algorithm is that a specific sample is an outlier. The boxplot visible in the figure 4.4 shows a visualization of the scores. It includes dotted lines indicating the standard deviations from the mean. The index of the points is also indicated for the points outside of the whiskers. For supervised algorithms, the locations of the real outliers are represented in red on the left side of the plot.

4.4.2 PCA projection

The results are also visualized on a 2 dimensional PCA projection of the dataset as illustrated in figure 4.5. The color goes from light blue for normal points to bright red for outliers. The points shaped as triangles represent the true outliers when displaying labeled datasets. The graph also indicates the variance captured by the first two components.

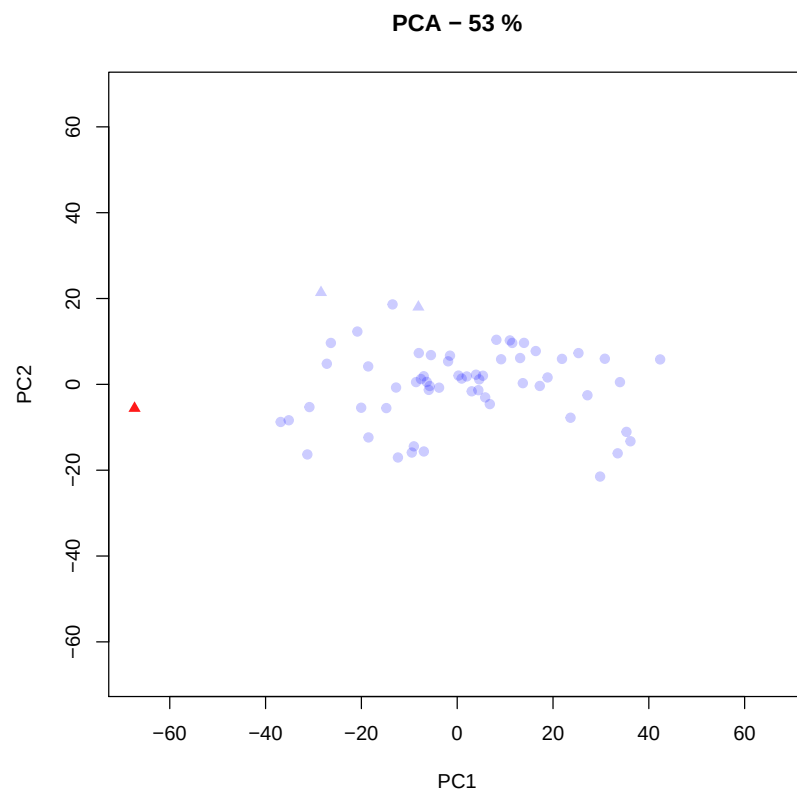


Figure 4.5: Example of the detection result displayed on a PCA. The red samples are the detected outliers. The variance captured by the two visible dimensions is indicated on the top of the graph

Chapter 5

Experiments

This chapter will present the results of different types of experiments based on simulated and real datasets. Section 5.1 will introduce the different type of datasets used in all the experiments. Section 5.2 will focus on the impact of different pre-processing transformations on the performances of the detectors in order to determine the best settings. Section 5.3 will compare the performances and stability of the different ensemble methods envisioned. Section 5.4 will present the detection performance of the different methods on a series of simulated and public datasets in a supervised setup while Section 5.4.2 will focus on the impact of removing outliers on the performances of a classification task.

We will close this chapter with an evaluation of the execution times for the different techniques based on the number of samples and the number of dimensions in 5.6.

The results presented in this Chapter will not be analyzed here. A separated chapter, Chapter 6 will be dedicated to the interpretation of those results.

The different boxplot figures presented in sections 5.2 and 5.3 have been sorted by means in order to improve the visualization and comparison of the performances. For the next sections, comparing the detectors, the order in which the detectors are presented is fixed to facilitate the comparison of the performances variation for a single detector across different experiments.

5.1 Experimental setup

5.1.1 Public datasets

The different public datasets we included in our results, commonly used in the literature, are all coming from the `Outlier Detection DataSets`[27]. Those datasets, however, presents a few flows:

- they are based on the assumption that the behavior of samples from a minority classes correctly approximates an outlier's behavior (see section 5.4.2 for a more detailed discussion);
- they do not exactly correspond to the types of datasets typically uploaded on REED as they all have many samples for only few features.

We included a presentation of the performances on those datasets as this assumption is frequently used in the literature and it seemed interesting to be able to examine how the detectors are performing on real data.

A representation of those datasets on two dimensional PCA projections are available on figure 5.1. Note that the colors used to plot them are from one of their best detectors.

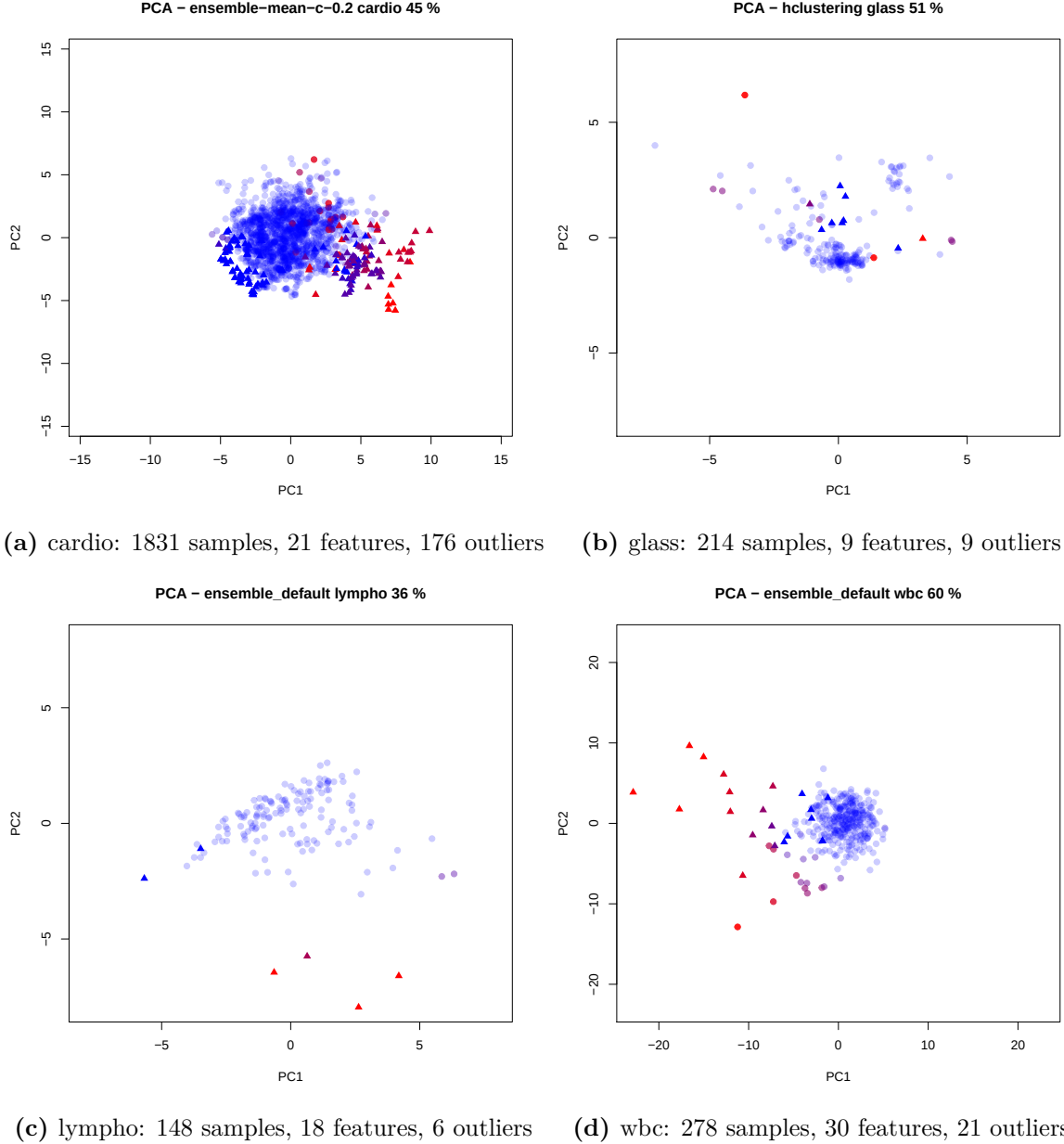


Figure 5.1: Pca of public datasets

5.1.2 Synthetic datasets

To perform those experiments in a controlled environment, we will also use two kinds of synthetic datasets in order to be able to assess the impact of induced variations on the performances of the detection methods:

r-files the first ones are composed of only one class, their features are multivariate-normal distributions computed with a **random** covariance matrix;

mc-files the second ones are composed of **multiple clusters**(two). The features of those datasets also follow multivariate normal distributions. They have been generated based on the means and covariance matrix of the most difficult dataset provided by DNAlytics **dataset1** to assess how the methods perform on more difficult datasets containing less information. The empirical covariance matrix learned from real data was not positive semi-defined. It has been corrected using the R method **Matrix::nearPD** in order to be

used to generate the datasets.

The variations will focus on:

the number of samples 50, 200 or 500;

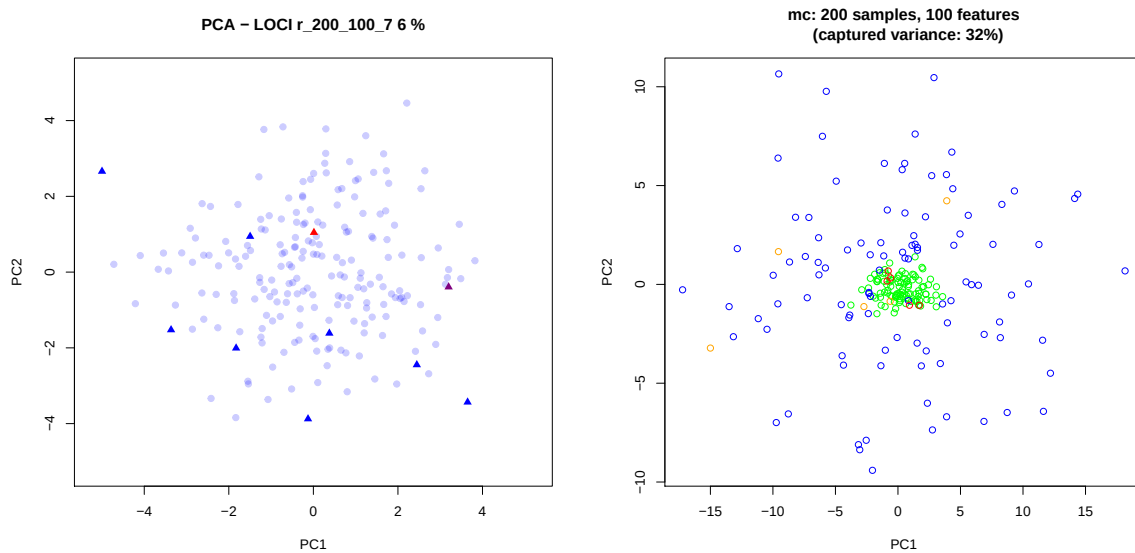
the number of features 10, 100 or 1000;

the number of classes 1 or 2.

As points located on the extremes of a distribution are less likely to be coming from that distribution (illustrated in figure 5.3b), we will consider those points, as outliers in order to establish the ground truth. We computed the mahalanobis distance using the real means and covariance matrix of the distributions and considered the data located in the 90 percentile of the dataset as normal points and the most extreme percentile as outliers. The gap between these two limits will be considered as noise and has been removed.

The advantage of using this technique, instead of introducing outliers manually is that it is more objective. Introducing outliers manually would require to define a behavior for those outliers, potentially leading to unrealistic introduction and over-optimistic results.

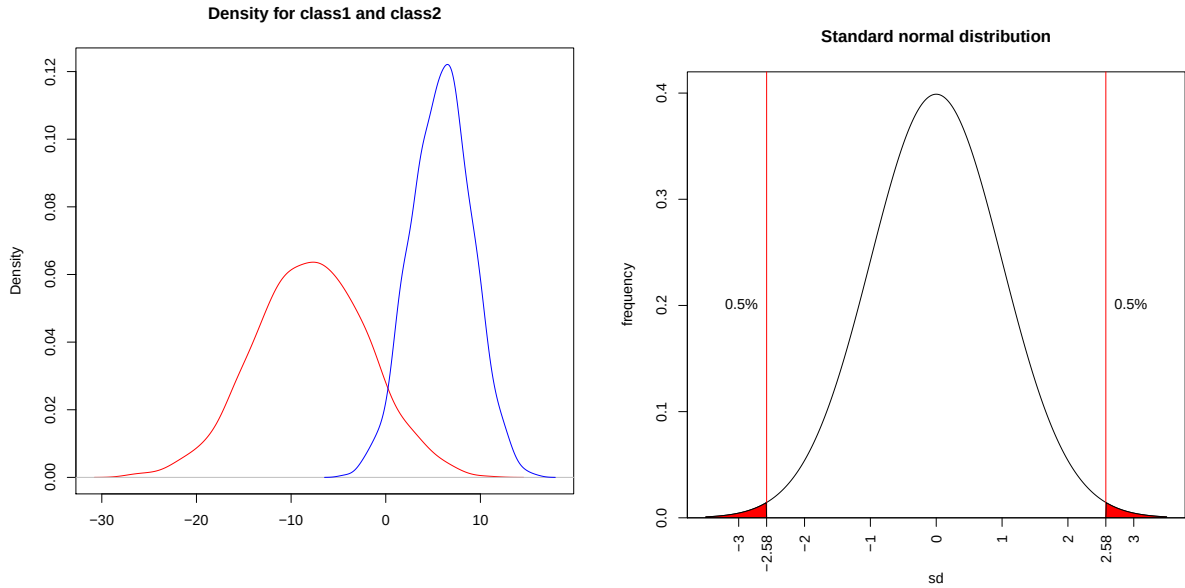
A PCA projection of a **r-file** and **mc-file** is visible in figure 5.2a and 5.2b. The color code in figure 5.2b is as follows : the two clusters are represented in green and blue with their respective outliers in red and orange. This representation allows us to visualize which points belong to which class more easily. As you can see on figure 5.2b, the separation of the two clusters is not visible on the PCA, which might indicate a poor cluster separation, as the kept variance is relatively high (32%). As illustrated in figure 5.3a, this overlap should have a negative impact on the performances when trying to detect the outliers at a global level.



(a) r files: 200 samples, 100 features, 10 outliers (b) r files: 200 samples, 100 features, 10 outliers

Figure 5.2: Pca of simulated datasets

For each of the different variations in size and dimension, we generated 10 different random datasets. In the **r-files** those datasets each have a random covariance matrix, in the **mc-files** a random resampling of the original dataset's feature has been used to compute the covariance matrices and means.



(a) Illustration representing the distribution of two overlapping classes. Considering the entire datasets, some outliers from one class might be considered as normal if they are in dense regions

(b) Illustration of a normal distribution with 1 percentile considered as outliers

Figure 5.3: Class distribution

5.2 Dataset transformation

The results presented in this section have been computed over four public files, nine¹ "r-files" and on nine "mc-files" with a global detection approach (without performing a detection per class).

As explain in the implementation chapter, multiple pre-processing steps have been investigated and in this section, we will try to evaluate which impact those pre-processing actions have on the quality of the outlier detection.

It is important to note that the results from the mutual information bagging have been excluded from all the results presented in this section except on figures 5.7 and 5.8. This bagging has been removed from all the other figures because the computation time was too long for huge datasets, preventing values to be available for them. This was giving the false impression that mutual information was showing higher results than it should have because the available numbers were only coming from smaller datasets.

This section is cut into two main parts, the first one is composed of the individual impact of each pre-processing methods on all detectors, while the second part is the opposite and looks the entire pre-processing process on a single detector at a time. As the three types of datasets are completely different, we have decided to present their results individually as it allows better discussion on the results.

5.2.1 Normalization

The boxplots visible in figure 5.4 are displaying the mean AUC over all the datasets computed by each base detector with the different pre-processing options (normalization, pca projection, bagging). It can be observed, on figures 5.4 and 5.5 that the impact is only significant in public datasets, and less important or unimportant in synthetic datasets.

¹50 samples and 10 features, 50 samples and 100 features, 50 samples and 1000 features, 200 samples and 10 features, 200 samples and 100 features, 200 samples and 1000 features, 500 samples and 10 features, 500 samples and 100 features, 500 samples and 1000 features

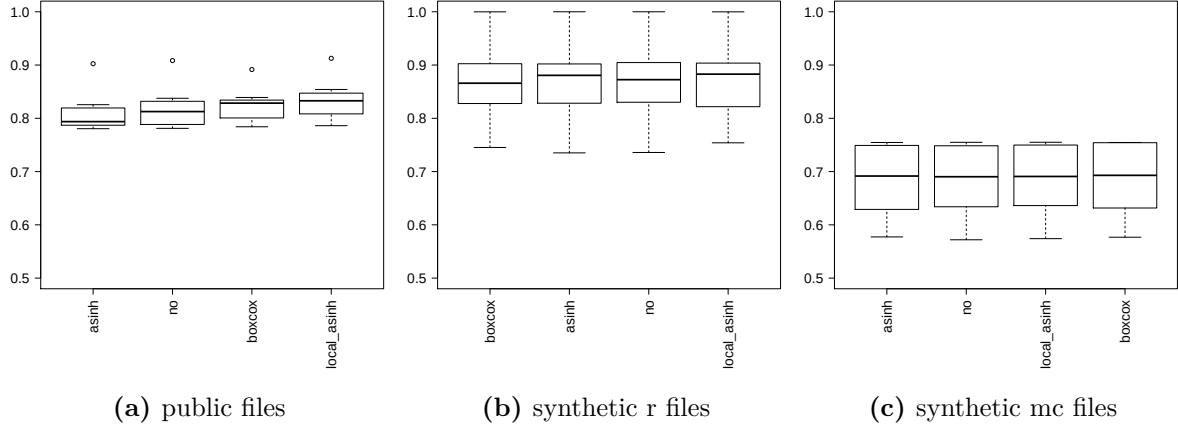


Figure 5.4: Impact of normalization on different types of input files

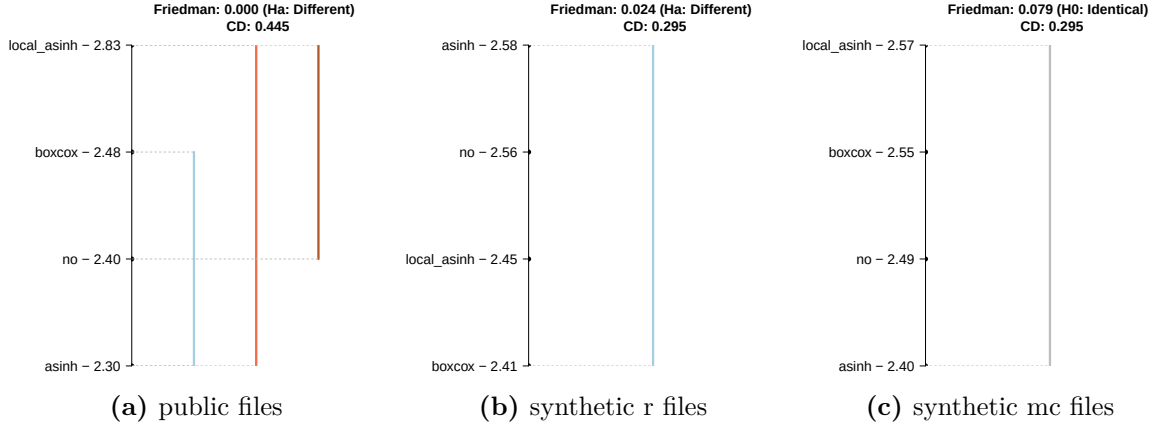
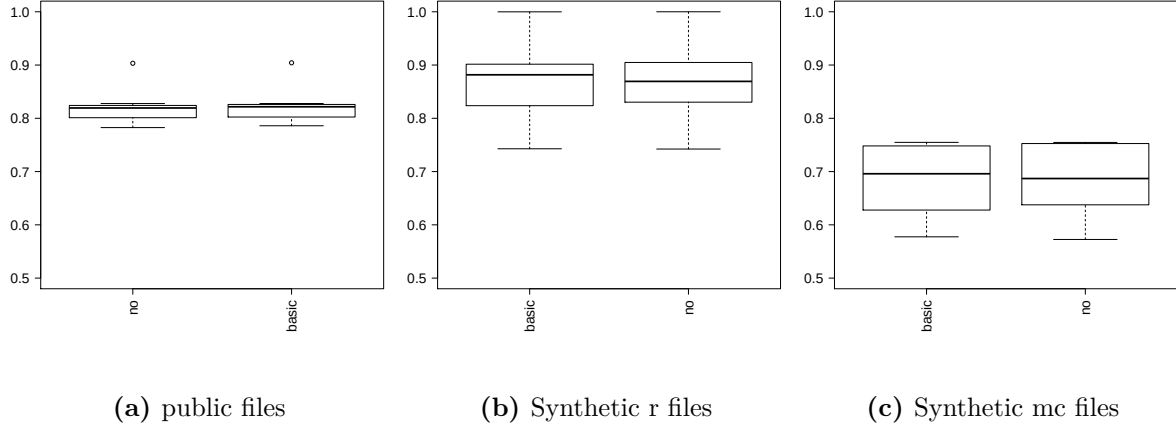


Figure 5.5: Friedman with Nemenyi post-hoc on normalization method.

5.2.2 Dimensionality reduction

5.2.2.1 PCA

The first dimensionality reduction test is based on the projection of the data on their PCA. The figure 5.6 displays the boxplots of base detector AUC (The AUC used is in fact the mean AUC of the detector on all the datasets) while changing the pca method. It can be observed, on figures 5.6 and 5.1 that the impact is weak on all the possible datasets. The paired T-test based on the rank display a small difference between them for the public and mc files but lead to different conclusions for the different datasets types. A paired t-test performed on the AUC directly is not detecting any significant difference.

**Figure 5.6:** Impact of applying a pca as pre-processing

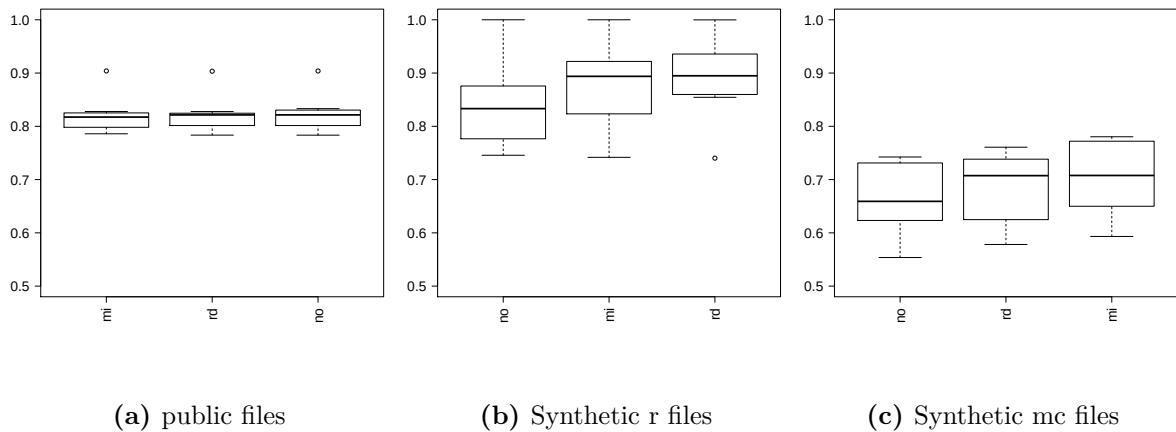
	p.value	conf.inf	conf.sup	mean		p.value	conf.inf	conf.sup	mean
public	0.524	-0.008	0.004	-0.002	public	0.012	-0.223	-0.027	-0.125
r	0.609	-0.005	0.008	0.002	r	0.315	-0.082	0.027	-0.028
mc	0.607	-0.004	0.006	0.001	mc	0.008	0.022	0.145	0.083
all	0.639	-0.003	0.004	0.001	all	1	-0.038	0.038	0

(a) performed on the AUC
(b) performed on the ranks

Table 5.1: paired t-test comparing not using a pca with using a pca for dimensionality reduction.

5.2.2.2 Feature bagging

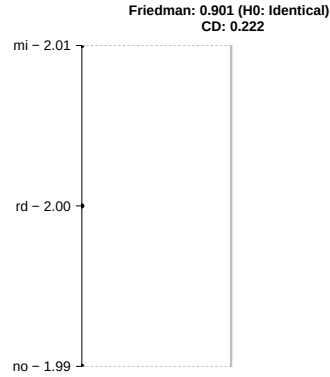
The second dimensionality reduction test is based on a feature bagging. We have experimented two kinds of bagging: one based on a random feature selection and the second one based on mutual information. The figure 5.7 displays the boxplots of the mean AUC per dataset for the base detector with different types of pre-processing while focusing on the bagging method. Unfortunately, as already explained in the introduction of this section, the high computational cost of mutual information prevented from using it on the biggest datasets. The results from mutual information have therefore been excluded from the statistical tests to avoid tampering the results. They have, however, been included, for reference, in the boxplot representation. As mutual information managed to handle all the public datasets, a Friedman test including mutual information has been conducted on those datasets only. This test, visible in figure 5.8, concluded to no difference. The paired t-tests performed on the two others are showing a difference in favor of random bagging compared to no bagging.

**Figure 5.7:** Impact of applying a feature selection bagging method

	p.value	conf.inf	conf.sup	mean		p.value	conf.inf	conf.sup	mean
public	0.414	-0.002	0.006	0.002	public	0.64	-0.07	0.043	-0.013
r	0	-0.061	-0.037	-0.049	r	0	-0.176	-0.062	-0.119
mc	0	-0.021	-0.01	-0.016	mc	0	-0.175	-0.051	-0.113
all	0	-0.032	-0.02	-0.026	all	0	-0.133	-0.061	-0.097

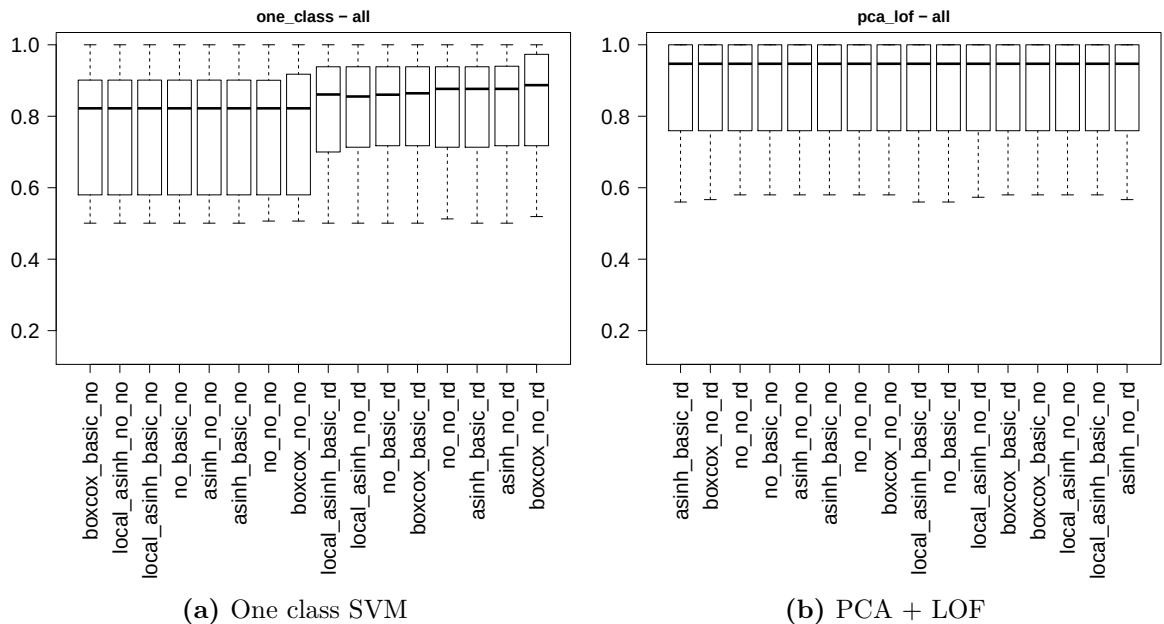
(a) performed on the AUC

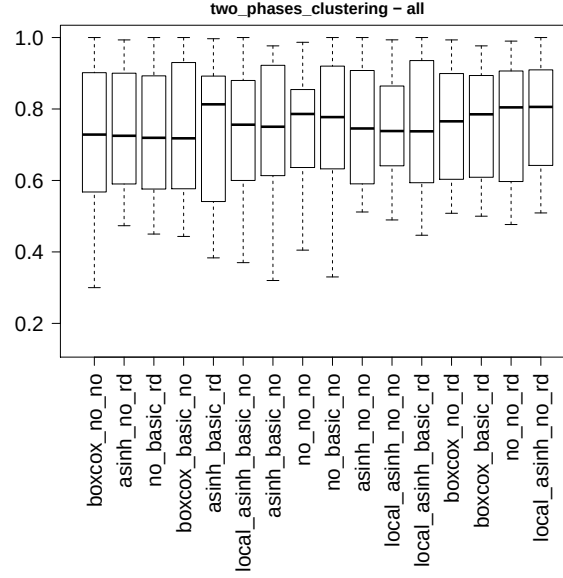
(b) performed on the ranks

Table 5.2: Paired t-test comparing not using a random bagging with using random bagging for dimensionality reduction.**Figure 5.8:** Impact of applying a feature selection bagging method on public files including mi

5.2.3 Pre-processing by detector

In this section, we will present the different configurations of pre-processing applied on individual detectors. Only a few interesting behaviors will be presented in (figure 5.10) but the results for all the detectors can be found in the annexes (in figures A.1 and A.2). The boxplots are sorted by their average mean; the boxplots have been computed over all the supervised datasets.





(c) Two phases clustering

Figure 5.10: Impact of the pre-processing on some base detectors

5.3 Ensemble

Even if performing the computation of the ensemble is the last phase of the detection process, we decided to include the experiments on the different ensemble methods before looking at the performances of the different detector in order to be able to include the best ensemble in the method comparisons of the next sections.

The experiments on the different ensemble algorithms described in the chapter state of the art and in the implementation chapter will focus on:

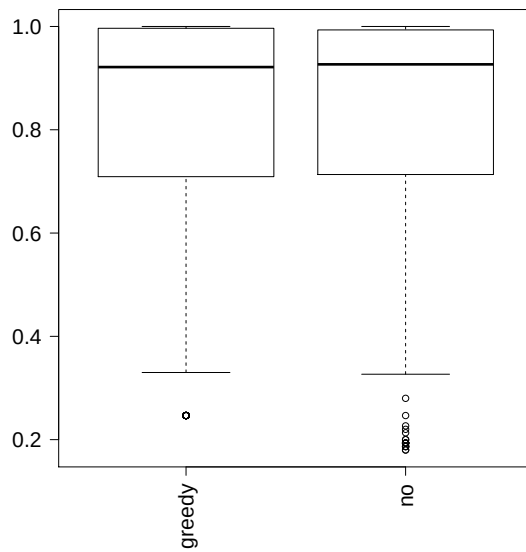
- performing a greedy base detector selection or not;
- the selection of the internal score composition function: average, median or maximum;
- adapting the weights of the scores based on a fix point(that use distance or common elements , with different sizes) or not.

In the following figures, b means basic which are the ensembles without fix point; c means that the common algorithm is used for the fix point; d means that the distance algorithm is used for the fix point. The numbers 0.2, 0.3 and 0.4 are the size used in the fix point.

5.3.1 Base detector selection

As the quality of the detector used to compose the ensemble will have an impact on how well the ensemble performs, we investigated the use of a dynamic greedy selection to discard ensemble having results too similar to each other and therefore increase the diversity of the results return by the ensemble.

The following results are based on performance comparison done using all the detectors to compose the ensemble or using only those selected using the greedy base detector selection algorithm. Those results can be found in figure 5.11, while the tables 5.3 display the results of t-tests made based on the AUC or on the ranking of the two methods (count of win and loss).

**Figure 5.11:** Greedy algorithm selection versus no selection

	p.value	conf.inf	conf.sup	mean
all files	0	0.003	0.005	0.004

(a) performed on the AUC

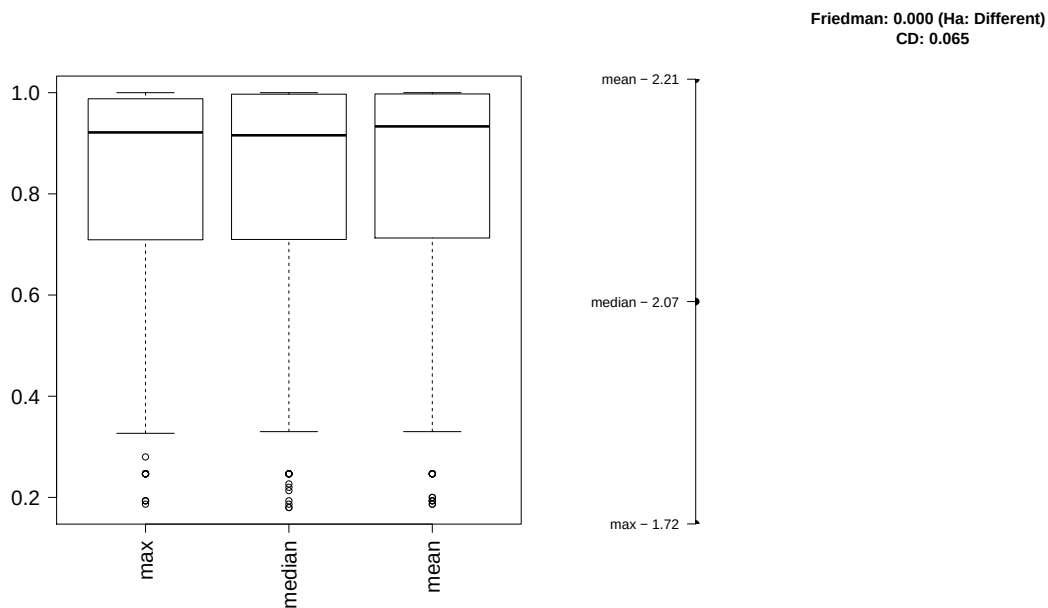
	p.value	conf.inf	conf.sup	mean
all files	0.171	-0.008	0.047	0.019

(b) performed on the ranks

Table 5.3: paired t-test comparing the entire base detectors with the detectors kept by the greedy selection algorithm

5.3.2 Internal function

We have tested the three most common algorithms, mean median and max as internal function of the ensemble. The boxplots of the AUC, made on all the datasets, using these functions, are present on figure 5.12a, while the figure 5.12b display the results of a statistic test comparing those methods.

**(a)** Internal function comparison for the ensemble**(b)** Impact of internal function in the ensemble**Figure 5.12:** Impact of the different functions

5.3.3 Base detector weight determination

The following figures report the results comparison from the test of the different weight determination algorithms, for the ensemble. The boxplots of the AUC are visible on figure 5.13a, while the figure 5.13b display the results from a statistical test between all these functions on the different datasets.

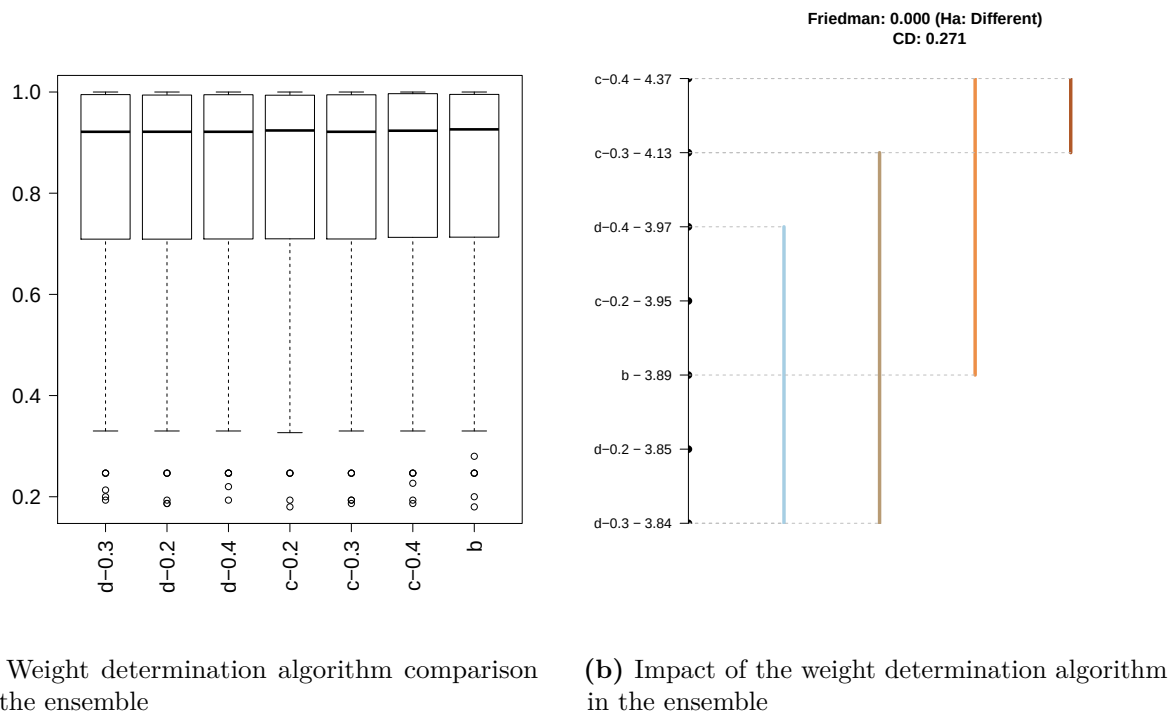


Figure 5.13: Impact of the weight determination algorithms

5.3.4 Global comparison

As you can see on the figure 5.14, the best ensemble, **ensemble-mean-c-0.4**, is the combination of all the best sub-parts from above. This ensemble will be the one used in the detector comparison of the next sections.

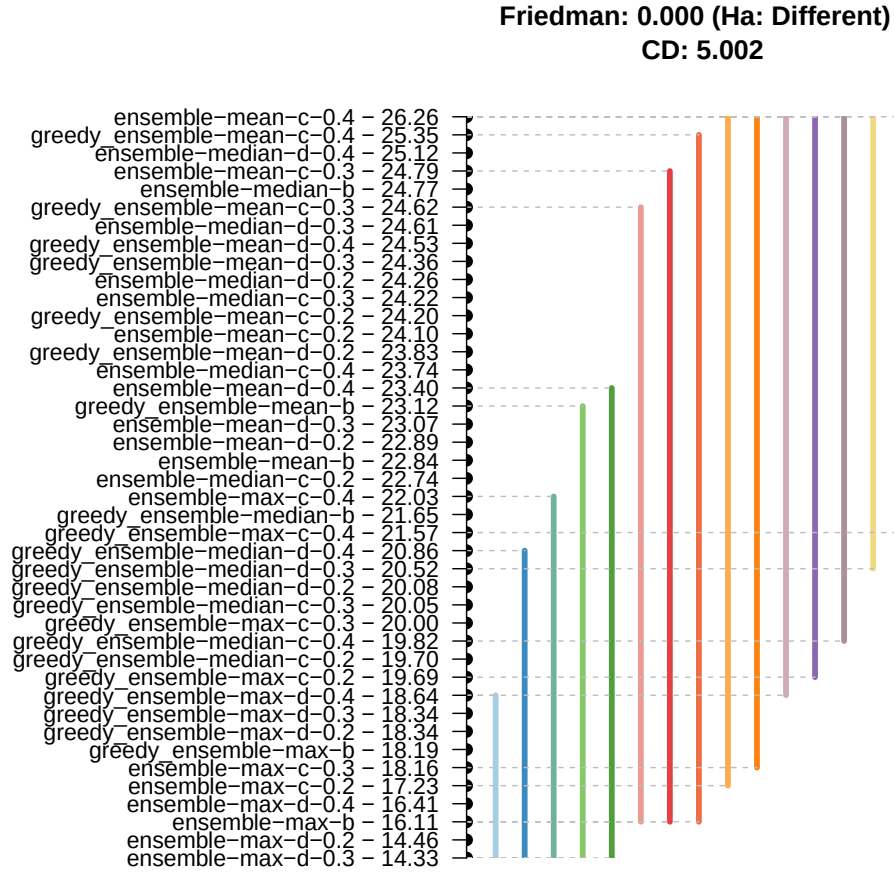


Figure 5.14: Impact of the ensemble algorithms

5.4 Supervised experiments

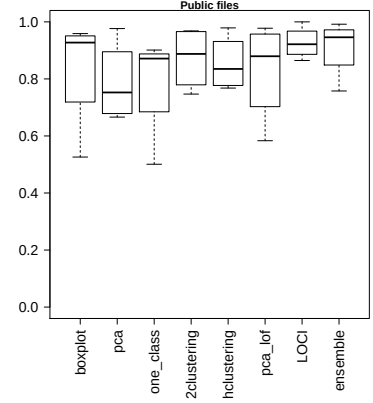
The following experiments we will be conducted using the local inverse hyperbolic sine for normalization, no pca projection and a random feature bagging. This choice will be explained in the result analysis in section 6.1.4. Those experiments are performed using 10 different randomly generated datasets for each of the tested samples(50, 200, 500) and dimensions (10, 100, 1000) sizes on the r and mc setup.

5.4.1 Base detectors

5.4.1.1 Public files

The following part will present the results computed on the public datasets using the minority class as outliers. The table 5.15a displays the AUC from every base detector and dataset while the figure 5.15b displays the boxplots (based on the four datasets) for every detector.

	Cardio	Glass	Lympho	Wbc
Boxplot	0.912	0.526	0.959	0.943
Pca	0.691	0.666	0.977	0.814
One class svm	0.874	0.501	0.869	0.901
2phases clustering	0.747	0.812	0.968	0.964
Hclustering	0.768	0.786	0.979	0.883
Pca_lof	0.583	0.822	0.978	0.937
LOCI	0.908	0.864	1.000	0.935
Ensemble	0.940	0.758	0.992	0.952



(a) AUC reporting

(b) Boxplots for every detector

Figure 5.15: Public dataset AUC analysis

No statistical test has been performed for those files due to the small number of datasets.

The table 5.4 displays the proportion of outliers captured considering a given threshold (0.01 to 0.1) of the dataset as outliers. The percentage indicated beside the dataset name is the real proportion of samples in the minority class. The **oracle** and **random** rows represents the proportion of outliers found if the detector was an oracle performing a perfect detection or in the opposite, selecting the samples randomly.

	Cardio - 9.6%			Glass - 4.2%			Lympho - 4.1%			Wbc - 5.6%		
Proportion	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
oracle	0.104	0.521	1.000	0.238	1.000	1.000	0.244	1.000	1.000	0.190	0.938	1.000
random	0.010	0.048	0.096	0.004	0.021	0.042	0.004	0.021	0.042	0.006	0.028	0.056
boxplot	0.080	0.341	0.540	0.000	0.111	0.111	0.167	0.667	0.667	0.143	0.524	0.619
pca	0.028	0.136	0.261	0.000	0.111	0.222	0.167	0.667	0.833	0.000	0.143	0.381
one class svm	0.080	0.239	0.392	0.111	0.111	0.111	0.167	0.333	0.333	0.095	0.333	0.524
2 phases clustering	0.034	0.426	0.580	0.000	0.000	0.222	0.167	0.667	0.833	0.190	0.571	0.714
hclustering	0.023	0.188	0.244	0.000	0.333	0.333	0.167	1.000	1.000	0.048	0.238	0.524
pca_lof	0.040	0.136	0.216	0.000	0.222	0.222	0.167	0.667	0.833	0.143	0.429	0.714
LOCI	0.095	0.278	0.500	0.000	0.000	0.333	0.167	1.000	1.000	0.095	0.429	0.667
ensemble	0.074	0.403	0.670	0.000	0.111	0.333	0.167	0.667	1.000	0.143	0.571	0.762

Table 5.4: precision at k: proportion of outliers captured while setting the detection threshold at 1%, 5% and 10% of the dataset

5.4.1.2 r files

Mono-class - number of samples and dimensionality variations In this section, we will report the performances of the different base detectors and of the best ensemble while varying the dataset size and the number of dimensions. The table 5.5 reports the mean AUC captured by every detector for each dataset size and dimension variation while the figure 5.16 displays the boxplots of the AUC for each detector, on all the datasets. The **n** and **d** rows in the tables are respectively indicating the number of samples and of features present in the datasets. The proportion of outliers present in each dataset is five percent.

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
boxplot	0.938	0.935	0.791	0.928	0.927	0.86	0.927	0.907	0.882
pca	0.989	0.703	0.619	0.994	0.979	0.599	0.999	0.998	0.684
one class SVM	0.742	0.773	0.917	0.981	0.816	0.981	1	1	0.987
2 phases clustering	0.932	0.955	0.868	0.97	0.842	0.773	0.891	0.822	0.655
hclustering	0.76	0.772	0.7	0.783	0.72	0.693	0.788	0.746	0.652
pca lof	1	1	0.993	1	1	1	1	1	1
LOCI	0.999	0.998	0.971	0.999	1	0.995	0.999	0.999	0.773
ensemble	0.999	0.996	0.945	0.999	0.997	0.986	1	0.991	0.991

Table 5.5: Table comparing the mean AUC of synthetic r files with a varying number of dimensions and samples

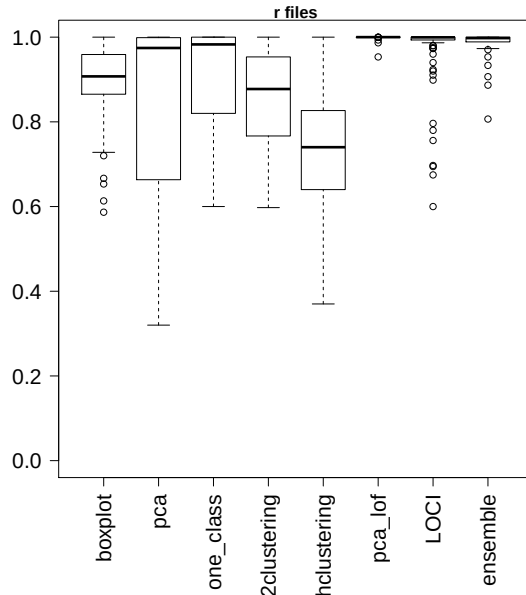


Figure 5.16: Boxplot of the results of the r files on the base detectors and the best ensemble method

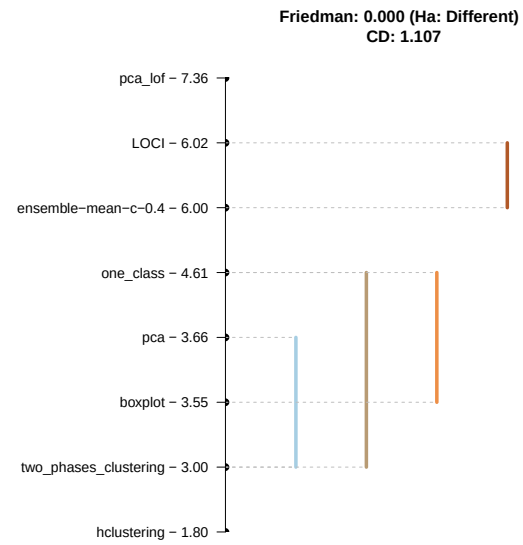


Figure 5.17: Friedman test with Nemenyi post-hoc test on the results of the r files on the base detectors and the best ensemble method

The figure 5.17 displays the results from a statistical test between all detectors on all "r" datasets. The tables 5.6, 5.7 and 5.8 displays the precision at k for each detector with k respectively equal to 1%, 5% and 10% of the dataset. The **oracle** and **random** rows represents the proportion of outliers found if the classifier were an oracle or a random function.

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	0.333	0.333	0.333	0.2	0.2	0.2	0.2	0.2	0.2
random	0.001	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
boxplot	0.3	0.3	0.133	0.19	0.19	0.12	0.2	0.18	0.148
pca	0.333	0.033	0.067	0.2	0.2	0.02	0.2	0.2	0.028
one class SVM	0.033	0.033	0.233	0.2	0.1	0.19	0.2	0.2	0.2
2 phases clustering	0.3	0.3	0.233	0.19	0.13	0.05	0.112	0.108	0.028
hclustering	0.133	0.067	0.1	0.15	0.11	0.05	0.16	0.084	0.072
pca lof	0.333	0.333	0.333	0.2	0.2	0.2	0.2	0.2	0.2
LOCI	0.333	0.333	0.167	0.2	0.2	0.2	0.2	0.2	0.176
ensemble	0.333	0.333	0.300	0.2	0.20	0.19	0.2	0.20	0.184

Table 5.6: Table comparing the precision at k with $k = n \cdot 0.01$ on of synthetic r files with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002
boxplot	0.7	0.6	0.367	0.74	0.57	0.37	0.788	0.528	0.432
PCA	0.8	0.1	0.167	0.9	0.74	0.07	0.948	0.916	0.136
one class SVM	0.1	0.067	0.633	0.81	0.23	0.74	0.996	1	0.676
2 phases clustering	0.6	0.667	0.4	0.67	0.34	0.22	0.412	0.312	0.144
hclustering	0.4	0.267	0.3	0.39	0.28	0.13	0.372	0.252	0.188
pca lof	1	1	0.9	0.99	1	0.99	1	1	0.996
LOCI	0.967	0.9	0.633	0.95	0.97	0.87	0.972	0.952	0.508
ensemble	0.933	0.900	0.667	0.940	0.910	0.700	0.996	0.760	0.812

Table 5.7: Table comparing the precision at k with $k = n*0.05$ on of synthetic r files with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.006	0.006	0.006	0.005	0.005	0.005	0.005	0.005	0.005
boxplot	0.9	0.767	0.533	0.87	0.71	0.52	0.864	0.7	0.564
pca	0.9	0.167	0.233	0.97	0.89	0.17	0.996	0.992	0.24
one class SVM	0.133	0.067	0.767	0.88	0.28	0.9	0.996	1	1
2 phases clustering	0.733	0.833	0.567	0.89	0.53	0.36	0.584	0.448	0.228
hclustering	0.467	0.4	0.333	0.5	0.38	0.27	0.496	0.36	0.272
pca lof	1	1	0.967	1	1	1	1	1	1
LOCI	1	1	0.9	1	1	0.98	1	0.996	0.592
ensemble	1	1	0.733	1	0.990	0.940	1	0.980	0.976

Table 5.8: Table comparing the precision at k with $k = n*0.1$ on of synthetic r files with a varying number of dimensions and samples

The boxplots of figure 5.18, displays the global evolution of the results of each detector while increasing the dataset's dimensions. Horizontally, the increase of the feature space (10, 100 and 1000 features) is studied. Vertically, the increase of the number of samples (50, 200 and 500 samples) is studied.

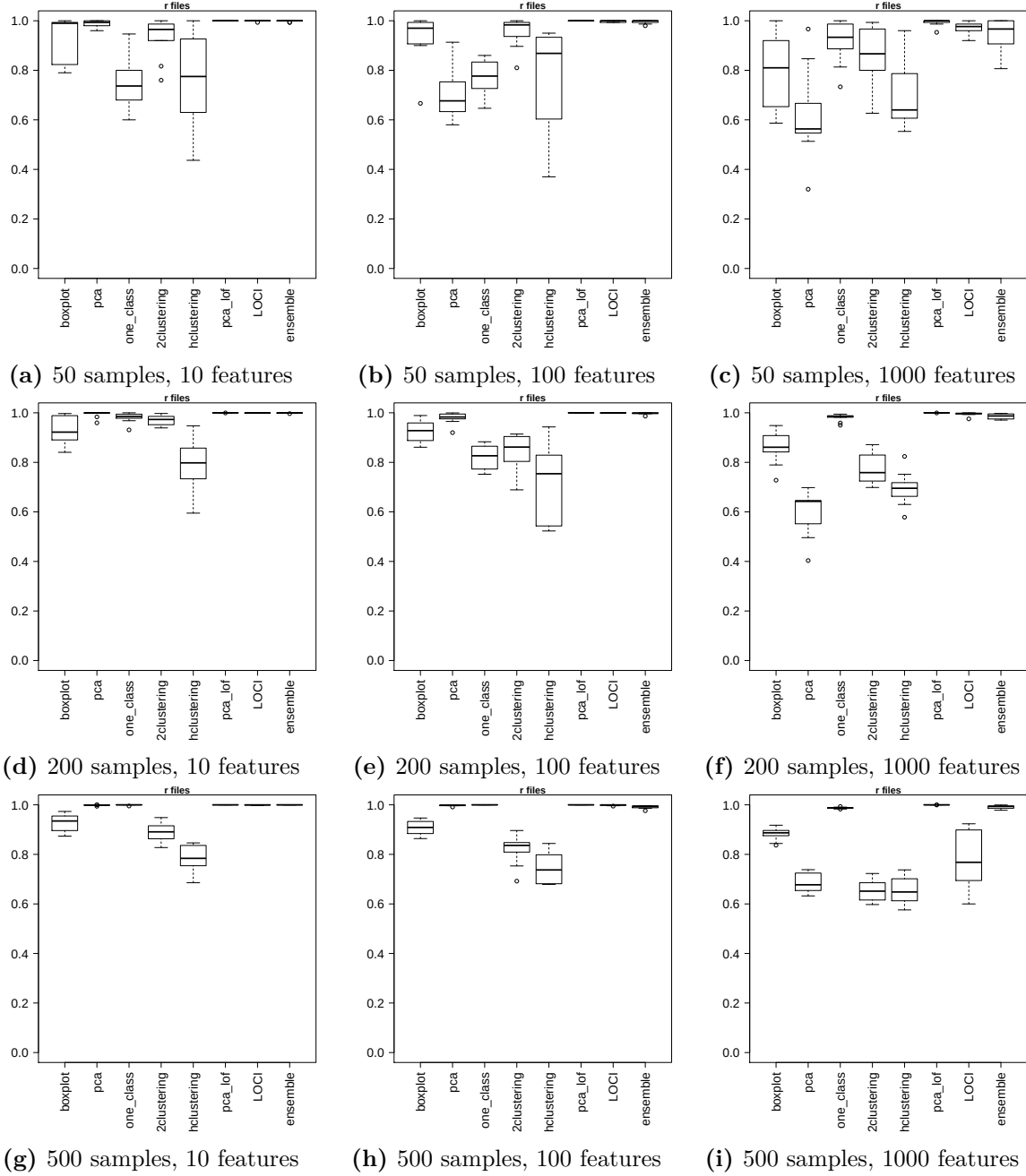


Figure 5.18: performances while varying dimensions

5.4.1.3 mc files

The results presented in this section have been computed on datasets containing two clusters. The performances are presented for each base detector and the best ensemble method and have been computed while varying the number of samples and the feature dimensionality of the datasets. The results displayed in the tables, are the means of the ten samples with the same dimensions. The **n** and **d** row table indicates the number of samples and the number of features present in the datasets. The proportion of outliers present in the datasets is five percent.

Global multi-class For those first experiments, we will perform the outlier detection on the entire dataset without giving any special focus on class of the samples. The class has been converted to a boolean variable and is considered as a normal feature.

The table 5.9 reports the mean AUC captured by every detector for each dataset dimension

while the figure 5.19 displays the boxplots of the AUC for each detector, on all the datasets. The figure 5.20 displays the results of a statistical test between all detectors on all datasets.

	50			200			500		
n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
boxplot	0.664	0.569	0.613	0.769	0.604	0.512	0.758	0.605	0.555
pca	0.827	0.515	0.596	0.922	0.678	0.578	0.921	0.684	0.597
one class SVM	0.772	0.695	0.602	0.903	0.698	0.559	0.932	0.755	0.596
2 phases clustering	0.725	0.639	0.621	0.831	0.629	0.559	0.752	0.59	0.528
hclustering	0.697	0.543	0.508	0.694	0.547	0.542	0.69	0.545	0.523
pca lof	0.83	0.63	0.586	0.942	0.735	0.573	0.941	0.811	0.611
LOCI	0.895	0.653	0.601	0.931	0.717	0.564	0.917	0.751	0.595
ensemble	0.836	0.626	0.603	0.908	0.688	0.566	0.912	0.716	0.600

Table 5.9: Table comparing the mean AUC of synthetic mc files with a varying number of dimensions and samples

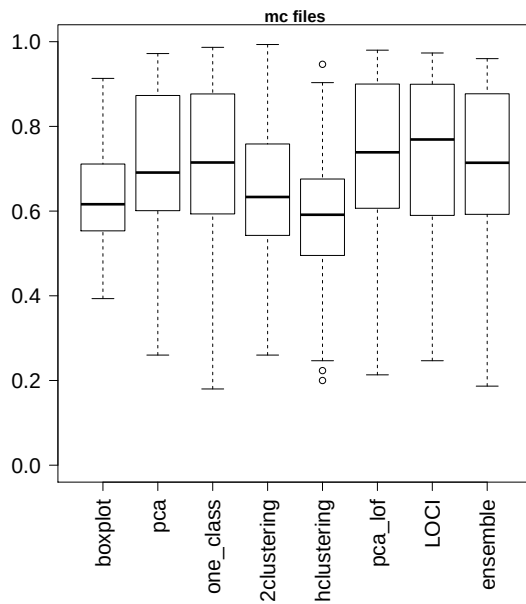


Figure 5.19: Boxplot of the results of the mc files on the base detectors

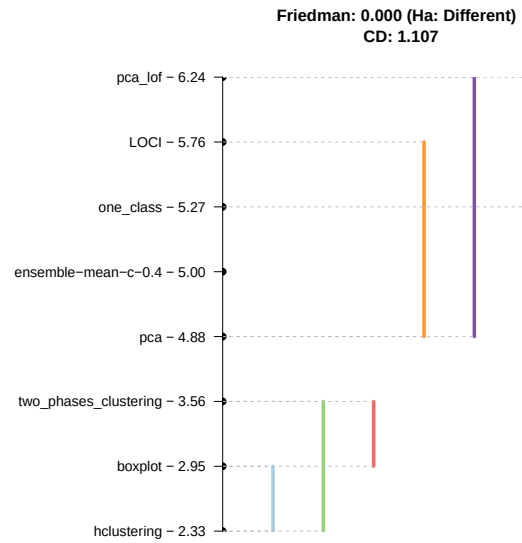


Figure 5.20: Friedman test with Nemenyi post-hoc test on the results of the mc files on the different detectors

The tables 5.10, 5.11 and 5.12 displays the precision at k for each detector with k respectively equal to 1%, 5% and 10% of the dataset. The **oracle** and **random** rows represents the proportion of outliers found if the classifier were an oracle or a random function.

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	0.333	0.333	0.333	0.2	0.2	0.2	0.2	0.2	0.2
random	0.001	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
boxplot	0.133	0.033	0.033	0.08	0.03	0.02	0.104	0.032	0.024
pca	0.1	0.033	0.033	0.18	0.04	0.03	0.172	0.056	0.016
one class SVM	0.067	0.033	0.067	0.17	0.04	0.03	0.184	0.088	0.008
2 phases clustering	0.2	0.033	0	0.14	0.02	0.01	0.036	0.036	0
hclustering	0	0	0.067	0.11	0.03	0.01	0.1	0.032	0.008
pca lof	0.167	0.033	0	0.2	0.08	0.04	0.196	0.084	0.024
LOCI	0.2	0.033	0	0.08	0.04	0.03	0.12	0.064	0.008
ensemble	0.233	0.033	0	0.11	0.04	0.02	0.116	0.044	0.020

Table 5.10: Table comparing the precision at k with $k = n*0.01$ on of synthetic mc files with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002
boxplot	0.2	0.1	0.067	0.25	0.07	0.06	0.276	0.112	0.068
pca	0.433	0.067	0.1	0.52	0.16	0.06	0.544	0.152	0.076
one class SVM	0.167	0.233	0.1	0.31	0.1	0.07	0.644	0.212	0.104
2 phases clustering	0.4	0.133	0.033	0.34	0.1	0.05	0.196	0.092	0.036
hclustering	0.2	0.133	0.133	0.28	0.11	0.07	0.228	0.08	0.04
pca lof	0.4	0.133	0	0.58	0.26	0.08	0.664	0.3	0.108
LOCI	0.4	0.133	0	0.4	0.18	0.07	0.468	0.164	0.076
ensemble	0.367	0.133	0	0.390	0.080	0.090	0.492	0.140	0.096

Table 5.11: Table comparing the precision at k with $k = n*0.05$ on of synthetic mc files with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.006	0.006	0.006	0.005	0.005	0.005	0.005	0.005	0.005
boxplot	0.367	0.133	0.1	0.37	0.1	0.11	0.38	0.164	0.128
pca	0.5	0.1	0.167	0.7	0.22	0.16	0.724	0.236	0.132
one class SVM	0.333	0.333	0.1	0.44	0.19	0.16	0.724	0.312	0.16
2 phases clustering	0.467	0.2	0.167	0.43	0.17	0.11	0.32	0.184	0.1
hclustering	0.267	0.2	0.133	0.34	0.16	0.1	0.356	0.144	0.112
pca lof	0.467	0.167	0.1	0.76	0.38	0.15	0.812	0.46	0.176
LOCI	0.6	0.233	0.133	0.66	0.28	0.12	0.72	0.304	0.152
ensemble	0.4	0.133	0.133	0.560	0.200	0.130	0.712	0.244	0.164

Table 5.12: Table comparing the precision at k with $k = n*0.1$ on of synthetic mc files with a varying number of dimensions and samples

The boxplots of figure 5.21, display the global evolution of the results of each detector while increasing the dataset dimensions. Horizontally, the increase of the feature space (10, 100 and 1000 features) is studied. Vertically, the increase of the number of samples (50, 200 and 500 samples) is studied.

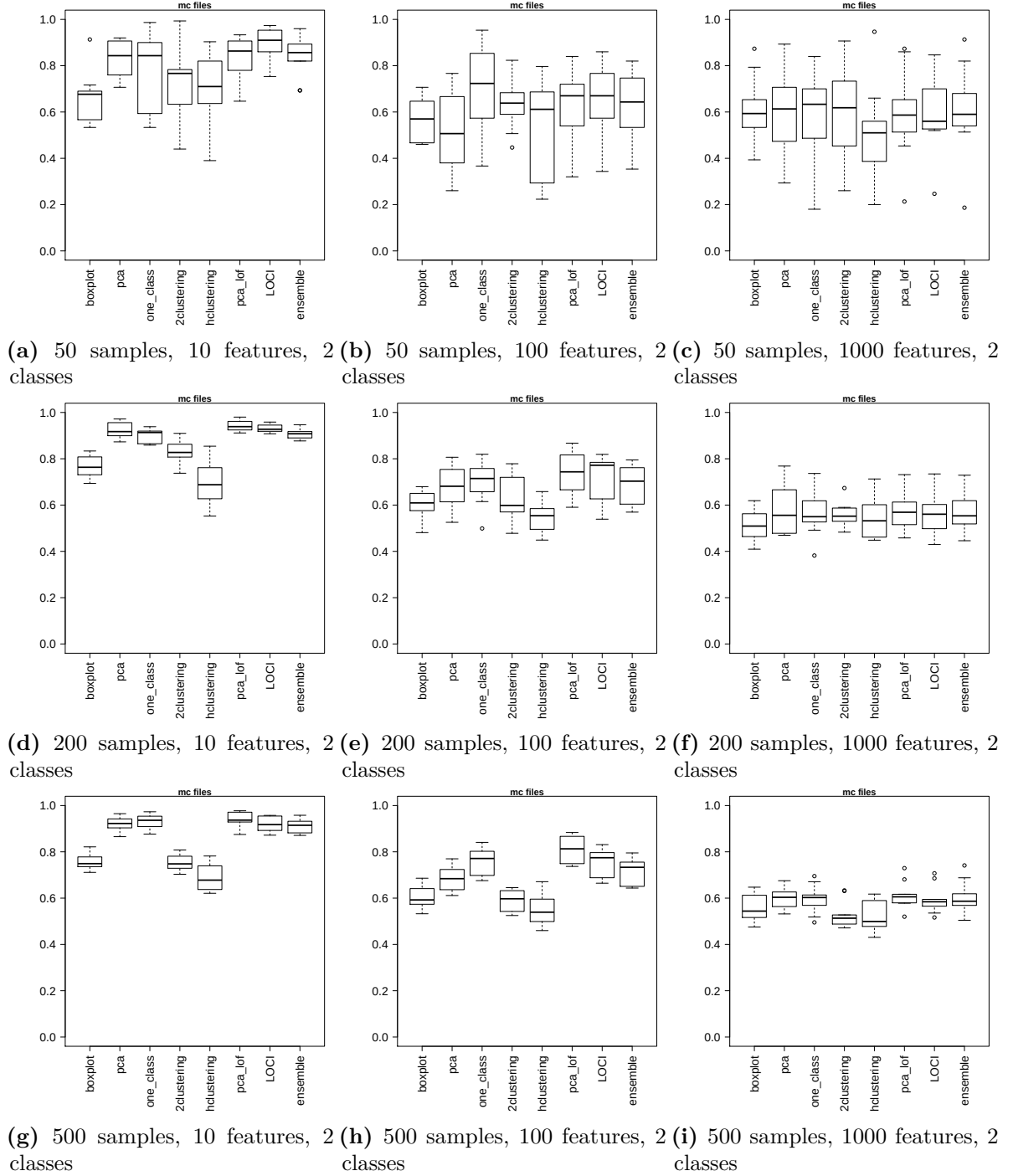


Figure 5.21: performances while varying dimensions

Contextual multi-class The results here under are performed on the same datasets than the previous ones but the detection is now done class by class and the detection results are assembled before the AUC are computed.

The table 5.13 reports the mean AUC captured by every detector for each dataset size/dimension variation while the figure 5.22 displays the boxplots of the AUC for by detectors, on all the datasets. The figure 5.23 displays a statistical test comparing the different detectors on all datasets.

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
boxplot	0.819	0.714	0.665	0.806	0.679	0.562	0.856	0.753	0.606
pca	0.723	0.551	0.606	0.934	0.594	0.55	0.94	0.699	0.579
one class SVM	0.688	0.603	0.556	0.838	0.682	0.566	0.952	0.786	0.604
2 phases clustering	0.58	0.525	0.592	0.901	0.67	0.564	0.863	0.689	0.566
hclustering	0.642	0.587	0.625	0.648	0.546	0.533	0.693	0.609	0.528
pca lof	0.926	0.7	0.661	0.974	0.724	0.577	0.979	0.814	0.613
LOCI	0.933	0.693	0.62	0.96	0.724	0.567	0.937	0.778	0.602
ensemble	0.934	0.722	0.650	0.963	0.718	0.566	0.968	0.794	0.605

Table 5.13: Table comparing the mean AUC of synthetic mc files, class by class, with a varying number of dimensions and samples

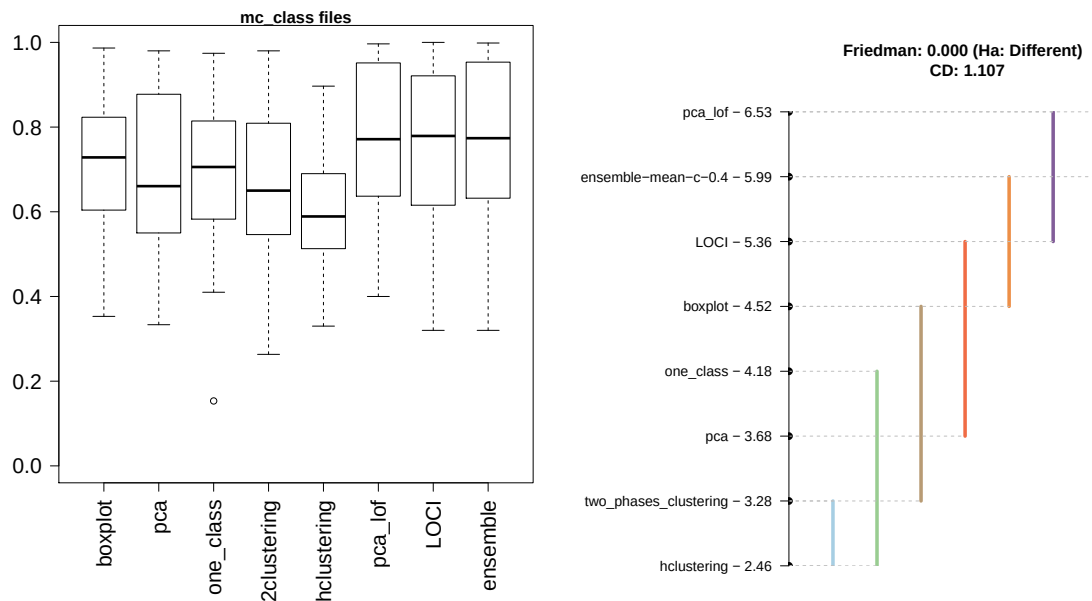


Figure 5.22: Boxplot of the results of the mc files, class by class, on the base detectors

Figure 5.23: Friedman test with Nemenyi post-hoc test on the results of the mc files, class by class, on the base detectors

The tables 5.14, 5.15 and 5.16 displays the precision at k for each detector with k respectively equal to 1%, 5% and 10% of the dataset. The **oracle** and **random** rows represents the proportion of outliers found if the classifier was an oracle on a random function.

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	0.333	0.333	0.333	0.2	0.2	0.2	0.2	0.2	0.2
random	0.001	0.001	0.001	0	0	0	0	0	0
boxplot	0.283	0.133	0.133	0.16	0.1	0.025	0.192	0.076	0.024
pca	0.233	0.1	0.133	0.155	0.055	0.03	0.138	0.058	0.016
one class SVM	0.05	0	0.167	0.04	0.005	0.01	0.218	0.096	0.032
2 phases clustering	0.167	0.083	0.1	0.15	0.085	0.035	0.142	0.07	0.04
hclustering	0.1	0.05	0.083	0.07	0.055	0.015	0.034	0.038	0.02
pca lof	0.283	0.15	0.217	0.195	0.11	0.02	0.22	0.102	0.056
LOCI	0.333	0.1	0.167	0.16	0.08	0.03	0.174	0.08	0.036
ensemble	0.283	0.150	0.167	0.175	0.100	0.020	0.196	0.082	0.018

Table 5.14: Table comparing the precision at k with $k = n \cdot 0.01$ on of synthetic mc files, class by class, with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002
boxplot	0.35	0.2	0.217	0.525	0.225	0.06	0.594	0.276	0.12
pca	0.25	0.117	0.217	0.44	0.135	0.065	0.37	0.176	0.062
one class SVM	0.067	0.017	0.183	0.14	0.07	0.07	0.664	0.226	0.1
2 phases clustering	0.217	0.117	0.15	0.495	0.22	0.09	0.428	0.188	0.112
hclustering	0.167	0.083	0.133	0.235	0.105	0.065	0.166	0.09	0.062
pca lof	0.433	0.2	0.283	0.765	0.315	0.09	0.806	0.35	0.148
LOCI	0.5	0.183	0.25	0.6	0.25	0.095	0.57	0.264	0.146
ensemble	0.400	0.200	0.283	0.680	0.290	0.070	0.704	0.290	0.118

Table 5.15: Table comparing the precision at k with $k = n \cdot 0.05$ on of synthetic mc files, class by class, with a varying number of dimensions and samples

n	50	50	50	200	200	200	500	500	500
d	10	100	1000	10	100	1000	10	100	1000
oracle	1	1	1	1	1	1	1	1	1
random	0.006	0.006	0.006	0.005	0.005	0.005	0.005	0.005	0.005
boxplot	0.617	0.417	0.317	0.69	0.33	0.13	0.754	0.414	0.242
pca	0.4	0.25	0.383	0.675	0.215	0.12	0.732	0.28	0.136
one class SVM	0.233	0.183	0.283	0.275	0.135	0.135	0.768	0.336	0.18
2 phases clustering	0.3	0.267	0.283	0.61	0.34	0.165	0.592	0.302	0.164
hclustering	0.2	0.183	0.25	0.32	0.145	0.165	0.258	0.176	0.144
pca lof	0.783	0.367	0.35	0.9	0.405	0.16	0.91	0.498	0.242
LOCI	0.733	0.4	0.333	0.85	0.37	0.145	0.78	0.398	0.232
ensemble	0.783	0.400	0.350	0.870	0.375	0.115	0.868	0.426	0.228

Table 5.16: Table comparing the precision at k with $k = n \cdot 0.1$ on of synthetic mc files, class by class, with a varying number of dimensions and samples

The figure 5.17 features a t-test comparison between the performances of the detector used on the entire dataset versus class by class.

	p.value	conf.inf	conf.sup	mean
AUC	0	-0.041	-0.022	-0.032
rank	0	-0.39	-0.242	-0.316

Table 5.17: T-test comparing the performances of the detector applied on the global level or per class. The first line is a t-test on the AUC, the second one on the ranking

The boxplots of figure 5.24, display the global evolution of the results of each detector while increasing the dataset dimensions. Horizontally, the increase of the feature space (10, 100 and 1000 features) is studied. Vertically, the increase of the number of samples (50, 200 and 500 samples) is studied.

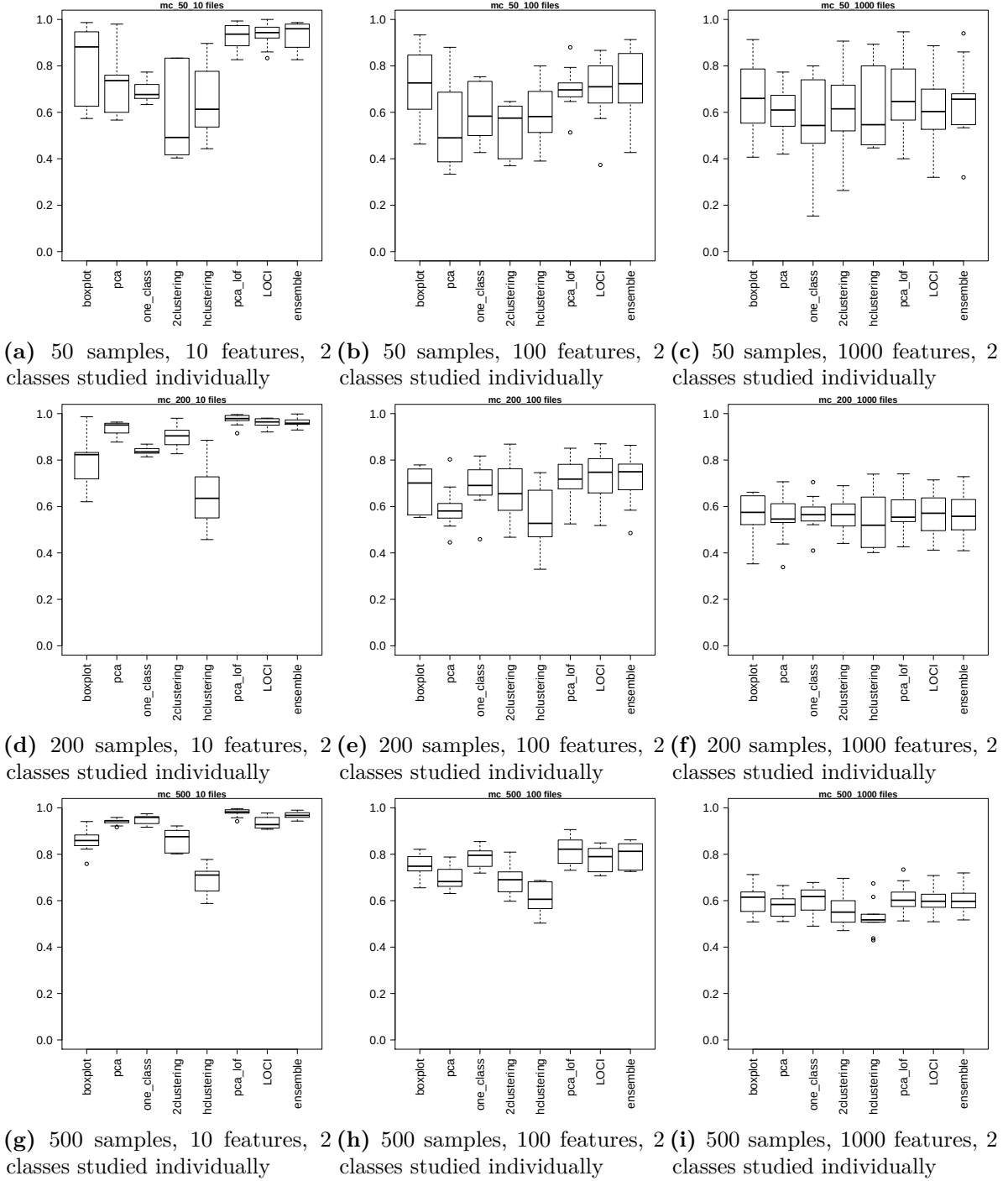


Figure 5.24: performances while varying dimensions

5.4.2 Unsupervised

In next experiments, the classification performances of two classifier algorithms, LDA and SVM, have been tested before and after removing outliers from the datasets. We introduced three different kinds of perturbations :

external the 1 percentile outliers present in the original datasets

shambles switching the classes of 5% of the samples in the dataset

extreme multiplying 10% of the features by a factor between 2 and 10 times their initial value for 5% of the samples in the dataset

The datasets are composed of two **r-files** (in order to have two classes), we ran 10 tests per dataset size/dimension variation, one for each of the random files, and using the next file as second class. For example, **r_500_1000_1** and **r_500_1000_2**. For each dataset, a 10-fold has been performed and the average result has been recorded, in the following tables and figures, the reported metric corresponds to :

original the classifier is trained and tested on the entire dataset without removing any outliers.

oracle_train the classifier is trained on a dataset which does not contain any outliers (removed using an oracle), the training set is still contaminated;

oracle the classifier is trained and tested on a dataset which does not contain any outliers (removed using an oracle);

predict the classifier is trained and tested on a dataset which does not contain any predicted outliers (using a 5% threshold)

Table 5.20 presents the average classification results for both classifiers, figure 5.26 the results for statistical tests on the different types of introduced perturbations and on all results.

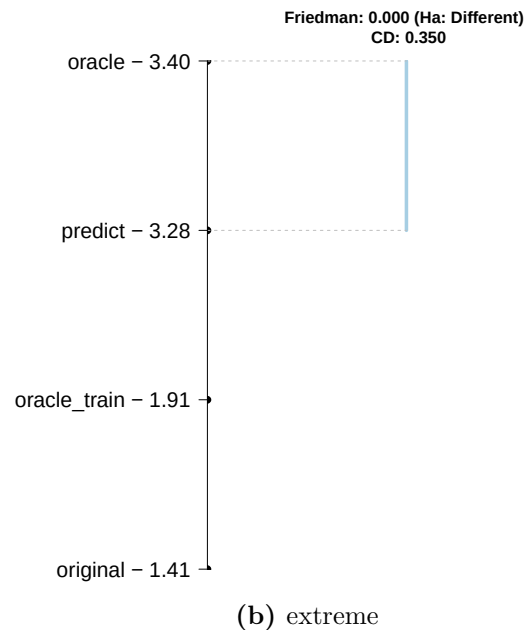
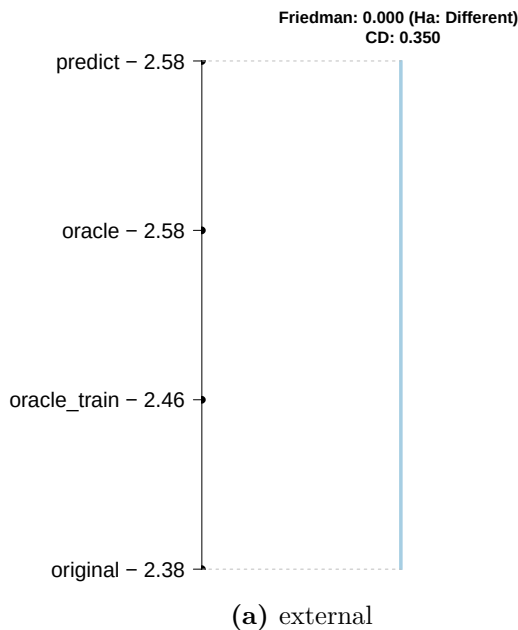
	original	oracle train	oracle	predict
external	99.984%	99.99%	100%	100%
shambles	95.013%	95.013%	100%	100%
extreme	97.83%	97.499%	100%	100%

Table 5.18: SVM

	original	oracle train	oracle	predict
external	99.862%	99.968%	99.989%	99.989%
shambles	90.721%	95.002%	99.988%	96.162%
extreme	92.692%	99.616%	99.972%	99.513%

Table 5.19: LDA

Table 5.20: Average accuracy



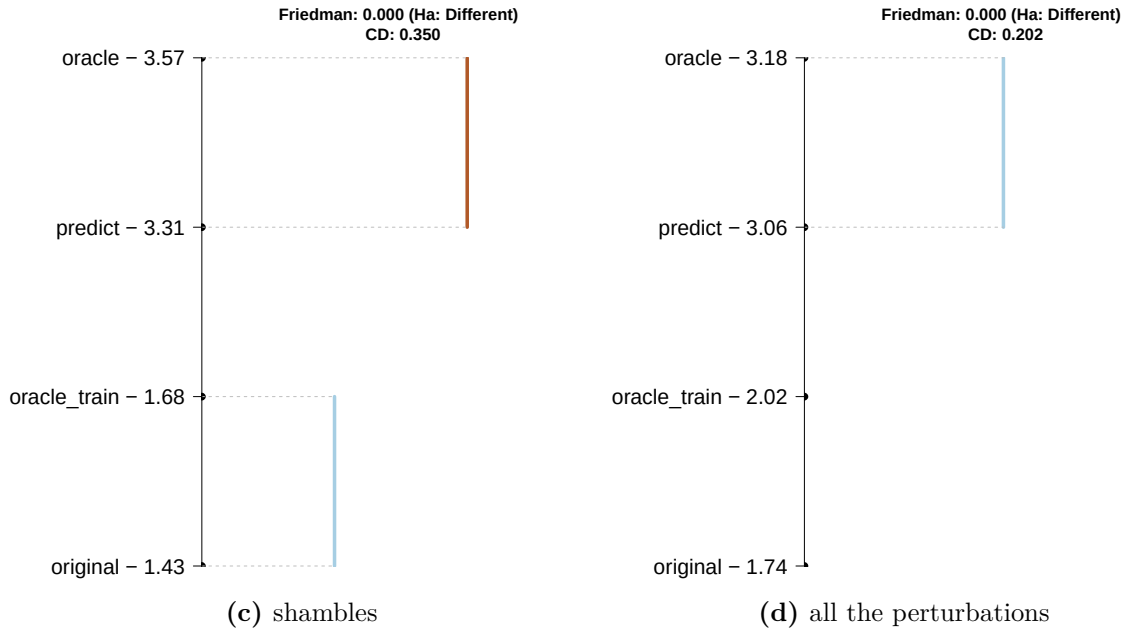


Figure 5.26: Friedman test with Nemenyi post-hoc test on different types of introduced perturbations

REED Table 5.21 presents the performances of REED’s classification before and after removing the outliers on the datasets provided by DNAlytics.

The column "% removed" indicates the percentage of the datasets that have been removed. Original is the original performance of REED on the full dataset, bad is the performances of REED when removing the samples that are the less likely to be outliers according to the detection scores, random is when removing random samples and predicted is when removing the samples detected as most likely to be outliers.

Dataset	% removed	original	bad	random	predicted
dataset1	10.4	55%	63%	43%	55%
dataset3	12.5	64%	61%	66%	67%
dataset4	12.9	98%	98%	98%	98%
dataset5	10.7	85%	86%	87%	83%
dataset6	7.5	77%	76%	78%	79%
dataset7	12.5	85%	83%	80%	88%
dataset8	13.2	75%	76%	78%	79%
dataset9	10.6	63%	71%	70%	63%
dataset10	5.7	83%	82%	83%	80%
dataset11	6.6	90%	91%	91%	91%
dataset12	9.7	96%	/	/	96%
mean	10.2	77.5/79%	78.7%	77.4%	78.3/80%

Table 5.21: Impact of the outlier detection on REED’s classification performances

5.5 Scores distributions

This section does not directly aims at presenting performance results but more at giving a graphical visualization of the shapes taken by the score distributions of the different algorithms. The figures in 5.27 presents a few interesting behaviors in a selected dataset `r_200_100_1`. The figures for all the detectors can be found in the annexes A.3.

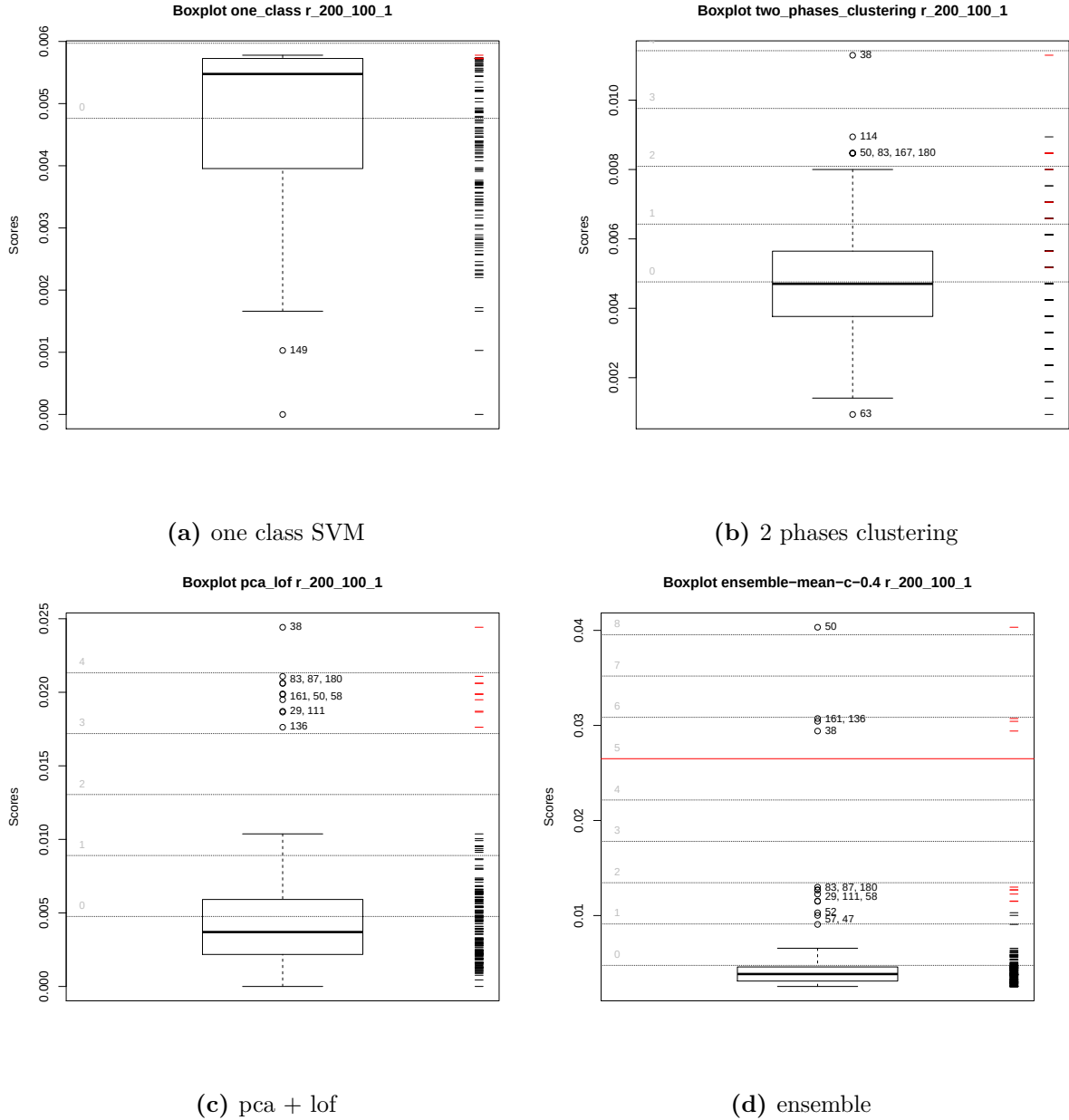
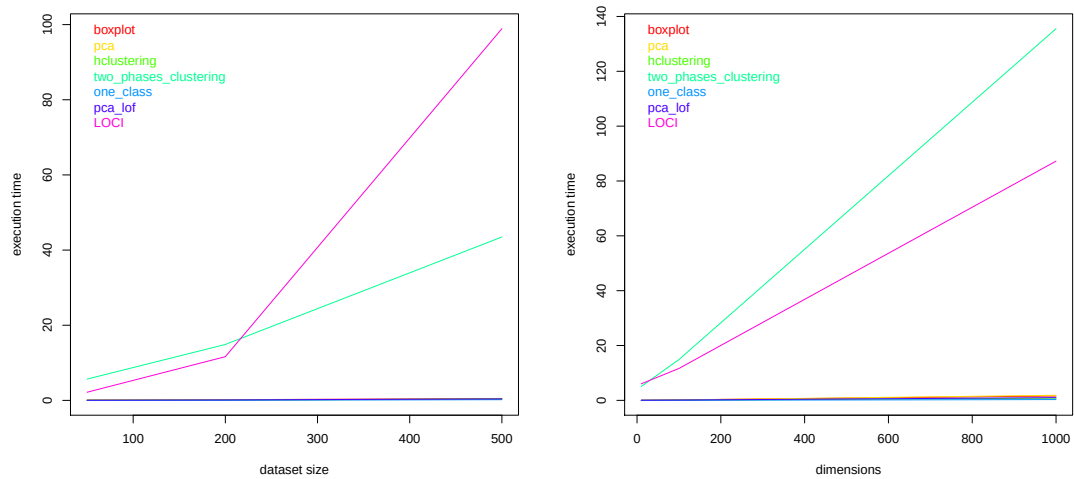


Figure 5.27: Boxplot visualisation of the scores returned by the different detectors

5.6 Computational performances

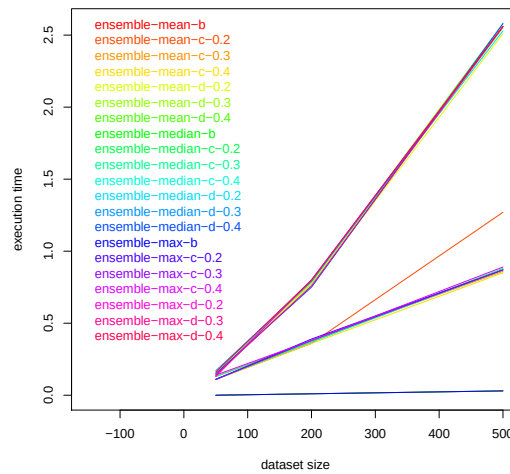
As this thesis aims at proposing a solution that can be implemented in REED, the computation time is also an important factor. This section will report the performances in terms of execution time for the different algorithms. As the ensemble methods are not impacted by the number of dimensions and as the scale of their computation time is much lower than for the base methods; they will be reported separately. All the graph of this section are present to show a tendency as we only consider 3 size variations, there are all made of straight lines, but the reader must keep in mind that it is only an approximation, and some of the detector, like for example the local correlation integral, do not run in a linear time.

Figure 5.28b and 5.28a are displaying the performances of the algorithms as a function of respectively the number of dimensions and the number of samples present in the datasets. for 5.28b, the number of samples has been fixed to 500. for 5.28a, the number of dimensions have been fixed to 100. Both figures are displaying results based on simulated datasets.



(a) Evolution of the computation type per method as a function of the dataset size

(b) Evolution of the computation type per method as a function of the dimension



(c) Evolution of the computation type per ensemble as a function of the dataset size

Figure 5.28: Performances evaluation

Table 5.22 is reporting the execution time in seconds for a selection of datasets and dimensions sizes for the different methods. Those measures have been performed on simulated datasets. The first two rows n and d indicate the number of samples and the number of dimensions.

d	10	100	1000	10	100	1000	10	100	1000
n	50	50	50	200	200	200	500	500	500
boxplot	0.01	0.06	0.54	0.03	0.19	1.6	0.08	0.47	4.53
pca	0.02	0.14	1.24	0.02	0.16	1.85	0.03	0.27	6.93
one class SVM	0	0.01	0.13	0.01	0.03	0.29	0.02	0.19	0.99
2 phases clustering	1.31	5.67	52.17	5.04	14.88	135.53	12.44	43.48	368.3
hclustering	0	0.01	0.12	0.02	0.07	0.79	0.09	0.39	5.7
pca lof	0.02	0.05	0.51	0.06	0.12	0.98	0.21	0.39	4.76
LOCI	0.59	2.17	20.86	6.03	11.63	87.24	69.57	98.91	382.24

Table 5.22: Table comparing the execution time of the different techniques on simulated datasets while varying the number of samples and dimension

Table 5.23 is reporting the execution time in seconds for the real datasets provided by DNAlytics for the different methods. The first two rows n and d indicate the number of samples and the number of dimensions.

d	2590	2590	165	88	48	48	26	12	11	10	7
n	48	64	194	112	40	40	303	132	628	501	31
boxplot	1.21	1.12	0.25	0.05	0.08	0.08	0.14	0.02	0.08	0.05	0
pca	3.07	2.43	0.36	0.07	0.1	0.17	0.03	0.02	0.04	0.02	0.01
one class SVM	0.33	0.3	0.08	0.01	0.01	0.01	0.01	0	0.02	0.03	0
2 phases clustering	100.56	101.11	17.54	4.03	2.47	4.42	6.7	3.03	10.88	10.44	0.67
hclustering	0.32	0.36	0.17	0.02	0.01	0.01	0.09	0.01	0.18	0.08	0
pca lof	1.26	0.79	0.19	0.04	0.04	0.04	0.18	0.04	0.35	0.19	0.01
LOCI	44.65	44.88	16.05	3.68	1.68	1.72	20.58	2.93	152.64	77	0.32

Table 5.23: Table comparing the execution time of the different techniques on the datasets provided by DNAlytics

Chapter 6

Results interpretation

In this Chapter, we will try to provide an interpretation, based on our knowledge of the different techniques and of the datasets, of the results presented in the Chapter 5.

We will start with a quick discussion of the impact of the normalization performed on the data and the dimensionality reduction in sections 6.1, will analyze the performances of the different ensemble techniques in section ?? and will finish with an analysis of the results of the different detectors in both supervised and unsupervised settings in sections 6.3 and 6.4.

6.1 Dataset transformation

In order to try to maximize the results of the base detectors, the first step is to supply them with the data in a form that increases their performances. This phase is done through the pre-processing and the feature bagging, which could be different based on the detector. Indeed, as the detector are based on different families of techniques, they don't especially require the same data transformation.

6.1.1 Normalization

As the mc and r files have been generated by using multivariate normal distributions, it comes to no surprise that the figures 5.4 and 5.5 does not display any significant improvement when applying a normalization step to the data.

We were not expecting any variation in the performances for those files and were, at first, surprised to see that there are minor variations in the performances visible in the r files. Those variations are, however, normal as :

- the asinh pre-processing is applied blindly on all the features without first checking if a normalization is required;
- the boxcox and local_asinh methods are only applied if the distribution appears not to be normal but evaluating the shape of the distributions is difficult in smaller datasets and this pre-processing might have been applied to some features;
- the two phases clustering detector is using random initialization which can also introduce some minor variations in the results.

On the public files, the difference in performances appears more clearly. As you can see in figure 5.4a, the normalization has a higher impact on the results. The worst one is applying the inverse hyperbolic sine over the entire dataset, which is to be expected as it will reduce the distance between the outliers and the normal points. The two best normalization methods are only applied on features with a non-normal distribution. The slightly better results obtained by the hyperbolic sine can be explained by the fact that the BoxCox transformation is performing

a more drastic transformation on the dataset (see figure 3.1 which displays an example of transformation) which might hide deviant behavior.

6.1.2 Dimensionality reduction

6.1.2.1 PCA

Looking at the boxplots in figures 5.6, applying a PCA reduction before performing the outlier analysis has little to no impact on the detection performances. Even though the results on the t-test indicates a significant difference in rank between the detection with or without pca (see table 5.1b) the same test performed on the difference in means (table 5.1a) indicates that even if indeed, one of the two methods is significantly winning against the other one, the difference between their scores are always really close and the improvement is negligible. As the t-test on the means and a t-test over all the datasets are not statistically significant, we can consider that the addition of a PCA is not providing a sensible difference on the performances of the detectors.

As a side note, as LOF is known in the literature to require a PCA to handle higher dimensionality, a PCA is already applied in our implementation of the method. This explains why LOF does not benefit from an additional projection.

6.1.2.2 Features bagging

The first clear visible result in the boxplots comparing the different bagging (figure 5.7) is that, in the figure 5.7a displaying the results on the public files, the bagging seems to have no impact at all. This is absolutely normal and is caused by the defined size of the bags. In those datasets, the dimensionality is not sufficiently high and the bagging is therefore not applied. The small difference in the results is due to the randomness of the detector two phases clustering.

As already mentioned, the results, that used the mutual information to create the bags, in the two other sets of files cannot be considered but it is interesting to see that the results of the different bagging methods are not positioned the same way on the r and mc files. This phenomenon is in fact due to the hardness of the datasets. On the r files, which are easier, the introduction of the bias does not have many impacts in small dimensionality and as it is not computable in high dimensionality, the random bags take the advantage. Otherwise, on the mc files, which are much harder, the bias has a huge impact even in small dimensionality, it takes too much advantage to let the NA in high dimensionality have an impact.

The results from the two other boxplots, comparing the AUC without bagging and with bagging are consistent for the two sets of simulated files. They both show an increased quality in the results when applying random bagging.

The paired t-tests visible in the figure 5.2 and the Friedman test performed on the public files in figure 5.8 are confirming this visual examination of the boxplots. There is no evidence of difference between the three methods in the public files and the test confirms that the increase in the results of random bagging is significant.

Even a really basic bagging algorithm such as random bagging allows the detectors to increase their results which is not a surprise when looking at how the high dimensionality affects the performances of most of the detectors (for example, see table 5.5 presenting the AUC from the different detectors on the r files), the added value of bagging is understandable as the curse of dimensionality is reduced when looking at a subset of dimensions. The combination of the results of the different bags then allows to detect which points are overall more confidently identified.

6.1.3 Per detector analysis

The results presented in the figure 5.10 are showing that the impact of the applied pre-processing varies depending on the base detector used.

The different parameterizations have, globally, the expected behavior, we can look at the impact on the base detectors. We decided to take the one with the most interesting results:

- **One class SMV:** the figure 5.9a shows almost no difference in performances when normalizing the data while the impact of bagging is clearly visible. This can be explained by the fact that, the more dimensions you have, the more you'll be able to find a separation between the samples (especially when you have more dimensions than samples). Reducing the number of dimensions therefore also reduces the overfitting;
- **Pca lof:** as you can see on figure 5.9b seems insensible to the pre-processing. As already mentioned, the pca has no impact in this case as one has already been applied in the detector itself. You can see that the only affecting factor, although with very limited impact, is the normalization which follows the global behavior: most of the boxplot with the best mean values are using the BoxCox or hyperbolic sine transformation applied only when the distribution appears not to be normal. This said, the differences are so small that they cannot be deemed significant;
- **Two phases clustering:** As you can see on figure 5.10c the noise is globally much higher and seems more random. It is totally true as this detector is the only one with a random part. This random impact seems much more important than the pre-processing on the data, in which it seems impossible to lead to conclusion as everything seems mixed, without the possibility to find a reasonable pattern.

In those figures, the impact of the normalization seems really low but no conclusion about its effectiveness can be drawn from those plots as the synthetic files, not highly impacted by the normalization, represents a big proportion of the results.

6.1.4 Conclusion

Overall, using a pca transformation as a pre-processing step is not improving the results for the selection of base detectors that have been tested. The feature reduction using random bagging seems to be the factor having the higher impact on the performances of the base detectors while the normalization of the data, applied when necessary, only has a positive impact on the public files which was to be expected.

In the following experiments, we will therefore apply the following pre-processing steps :

- **local_asinh** to normalize the features that seem to follow an exponential distribution;
- **no pca** as it does not improve the quality of the results and increases the computation time;
- **rd (random bagging)** as it consistently shows a small performance improvement.

6.2 Ensemble

To evaluate the ensembles, we have only done a global analysis, without taking into account the type of datasets used. This decision comes from the fact that the ensemble must be as robust as possible and truly independent of the type of dataset.

As a reminder:

- **ensemble-b** contains all the ensemble that have fixed weight for each base detector;
- **ensemble-c** contains all the ensemble that used the **common** algorithm to determine the weights for each base detector;

- **ensemble-d** contains all the ensemble that used the **distance** algorithm to determine the weights for each base detector;
- **ensemble-greedy** contains all the ensemble that used the **greedy base detector selection** algorithm.

6.2.1 Base detectors comparison

The first important decision, when talking about ensembles is the choice of detectors composing it. In section 5.3.1, we compared the results of ensembles using either the complete set of detectors or only a subset of the detector computed dynamically using the greedy model selection described in algorithm 8.

The boxplots of figure 5.11, sorted by means, seems to indicate a really small tendency for the complete set of detectors to have better results than their greedy version but a t-test performed to compare their performances and visible in table 5.3 shows that the difference is really small. If comparing the ranks (which algorithm outperformed the other per dataset) no significant difference is detected while a very small difference is detected in favor of not using a greedy selection when comparing the AUC directly.

We can conclude from these two tests that applying the greedy algorithm does not improve the quality of the results and even tend to slightly downgrade them, which is in opposition to the theory. This phenomenon can be explained by the fact that the fix point, discussed in the next section, is more beneficial and is negatively impacted by the greedy selection. The purpose of the greedy algorithm is to choose detectors leading to different results, which therefore reduces the redundancy in the results used by the fix point to adjust the weights.

6.2.2 Internal functions comparison

The second thing to consider is the choice of internal function. The internal function computes the scores given by the ensemble based on the scores given by the base detectors.

We have considered the assembling functions most frequently used in the literature, the average, the median, and the maximum. The boxplots of figure 5.12a, sorted by means, seems to indicate a small performance increase when using the average.

The statistical test confirms that the difference between the three methods appears to be significant as the Friedman p.value is equal to zero and the difference in average rank for each method is greater than the Nemenyi critical distance. The literature also favors averaging as it is the one that reduce the most the variance. Conversely, the maximization, is the less recommended one, in agreement with this Friedman test, as its effect is clearly more random and thus has the highest variance.[1].

6.2.3 Base detectors weights

In addition to the internal function a weight can be added to the scores returned by the base detectors to increase the impact of the ones that seems more likely to provide the best results.

To choose those weights, the ensemble executes a fix point, with two possible functions to compute the weights:

- the first one is based on the number of common elements in the top of the base detector results and the top elements of a target vector (**ensemble-c**);
- the second one is based on the distance of the top elements of a target vector and the results of the base detectors (**ensemble-d**).

The target vector of the two methods is in fact the ensemble computed with the previous weights, while the top elements are the twenty, thirty or forty percent of the samples with the highest outlier scores.

No visible difference can be seen by looking at the boxplots of figure 5.13a. As they are sorted by means, it seems that the average results of the method not using any weight are slithery better.

The Friedman test performed to see if any statistically significant difference can be exposed and visible in figure 5.13b reports that the functions are different as the p-value is equal to zero. This might look surprising as the algorithm **b** had the better means on the boxplot but the Friedman test is based on the ranks between the algorithm while comparing the means can be biased by a sample with a higher results. The common algorithm with a top proportion of forty percent wins. Even if the benefits of the fix point was predictable, taking the biggest proportion is much a surprise as we want to limit the impact of the inliners. However, this large proportion must be caused by the fact than the base detectors come from very different family, and thus, find different outliers. A higher proportion of points therefore has to be considered.

6.2.4 Ensemble: conclusion

The statistical test performed on the different configurations in figure 5.14 shows that the parameter with the higher impact seems to be the internal function as all the ensembles using the average are in the top (the fourteen instances are in the first twenty propositions), while all the instances using the maximization are in the bottom (the fourteen instances are in the last twenty-one propositions). However, this test clearly goes in the same direction as the individual analysis, as the best ensemble, **ensemble-mean-c-0.4**, is in fact the combination of the three best parameters previously identified.

6.3 Supervised results

6.3.1 public files

The results presented in the figure 5.15a are interesting as they go in the direction of our main assumption: when working with different types of datasets, it's not always the same detector which is the most powerful. Even if we could not perform a statistical test on those datasets, we can observe some tendencies in the results.

LOCI is doing a really good job on all the datasets and has the smallest variance (see the boxplots in figure 5.15b).

Globally the results are not too bad, except on the dataset glass where some of the detectors, such as boxplot and one class SVM really struggled to predict the outliers correctly but looking at the PCA projection in figure 5.1b, the outliers are located in the middle of the PCA for the two main components. Looking at the results of the precision at k table for glass (table 5.4), we can see that even if the AUC for the other methods are not as bad for this dataset, they all failed to return the outliers in first results, performing even worse than if selecting the sample randomly.

Even if the boxplot method has a big variance due to the bad results on glass, its results on the other datasets are really good. Its detection is based on the extreme value analysis of individual features which means that it is well suited to find outliers having extreme value on a per feature basis. Its good performances here can probably be explained by the assumption that the minority class can be compared to outliers. If the dataset contains a lot of significant features for classification, the boxplot method will perform well as it will identify deviant behavior for all those features. On the opposite, if there is an overlap on the distributions of the two classes, the "outliers" will be impossible to detect.

This shows that those results really need to be interpreted with caution as the main assumption in using them does not really correspond to the reality of what an outlier is.

Even though the results are really poor for some methods, the ensemble manages to always give, if not the best, at least one of the best results which is logical given the stabilizing factor it

provides. On the dataset cardio, it even manages to outperform the results from the individual detectors.

6.3.2 Synthetic files

6.3.2.1 r files

This type of dataset is really straightforward and we were expecting really good performances on it, especially from PCA as the decision to consider a point as an outlier is based on the Mahalanobis distance on the real distribution. So even though it is a really basic detector, we were at first surprised to see that it did not perform well at all. Looking at table 5.5, the performances drastically deteriorates when the number of samples is smaller than the number of dimensions while its performances are much better otherwise. Our understanding is that this issue is caused by the curse of dimensionality, flattening the distances and therefore reducing the difference between normal points and outliers added to the fact that the empirical means and covariances does not exactly correspond to the ones of the original distribution.

Overall, we can really see the impact of increasing the dimensionality on the performances of most detectors, interestingly, as you can see on the figures 5.18, increasing the number of samples does not have a big impact on the detector's median performances but more on the variance of the results. We can see that the boxplot tends to be more concentrated around their median values.

The drop of performances visible for LOCI on the last figure (5.18i) can be explained by the way LOCI computes its neighbors. When the dimensionality becomes too big, it fails to detect any neighbors and tend to return the same score for every point. The reason why this problem does not arise for smaller datasets is because of the bagging performed on them which reduced the dimensionality sufficiently. But as the random bagging is based on the number of samples, with 500 samples too many dimensions are kept for the algorithm to perform correctly. Note that this problem is less present on real datasets as the distances between the points are less uniform. This can seem really bad but, in practice, when the algorithm obtains really low performances due to its inability to correctly detect any outliers, the scores it gives are really similar for all points so it does not negatively impact an ensemble using its results.

Without surprise seeing the results, the statistical test identifies LOF as being significantly superior to the other techniques, followed by LOCI and the ensemble method. The very good results of LOF and LOCI need to be considered with caution as we are dealing with a synthetic dataset that apparently really favors them, such perfect performances cannot be expected when working on real datasets.

Looking at the tables displaying the tables displaying the precision at certain thresholds 5.6, 5.7, 5.8, while a lot of detectors managed to get the maximal number of detectable outliers when considering the 1% threshold (meaning that they managed to detect real outliers in their first results), the ones having difficulties don't really improve their performances when increasing the threshold, meaning that they completely failed to detect the missing real outliers and are confusing them with normal points.

6.3.2.2 mc files

Global Those datasets are more complex than the ones from the previous experiments. The two clusters have really close means as you can see on the pca projection on figure 5.2b and the covariance matrix of the features contains less information. This will make the, global, detection harder as the outliers of one class could perfectly lie in the other class. In comparison with the "r-files", two, expected, things are clearly visible in the table 5.9 and in the figure 5.19:

- the results are far worst, the means are lower (68% versus 89%) while the variance is much higher(0.015 versus 0.009);

- the two Friedman tests, figure 5.17 and 5.20, are pretty similar. The detectors are seen as different, the p-values and the confidence intervals are the same. The ranking of these detectors is almost the same, the local outlier factor and the local correlation integral are far better than the others, while the hierarchical clustering is the worst.

The last analysis concerns the precision at k , reported on tables 5.10, 5.11 and 5.12, which follows the above tendency of being less good than the results of the "r-files". However, the comparison of the precision at k with the results of the random detector shows that even if the mean AUC in high dimension is close to 60% for all detectors (figure 5.21), which is not too far from a perfectly random detector where the AUC would have been 50%, the precision at k is far better than the random detector.

Which shows, once more, than the AUC is not the only parameter to look at: what is important in outlier detection is to find the outliers as quickly as possible.

The high dimensionality problem is clearly visible, in figure 5.21, with these datasets. The mean AUC of every detector concentrates around sixty percent. This figure, also perfectly displays the score concentration while increasing the number of samples. However, this concentration only leads to a performance increase in small dimensions; in high dimensions, the mean result remains almost identical.

By class Comparing the results in the table 5.13 with those of table 5.9, there is a visible gain in detection performance when performing the detection class by class instead of at a global level. We were, however, surprised by how low this performance increase was as we were expecting better performances when taking the class into consideration. Our interpretation of this reduced impact is that the class is not totally ignored in the global detection as it is converted to a boolean variable, it therefore contributes to the global detection as well.

The t-test from table 5.17 comparing the average performance of the detector class by class and on the entire dataset confirms that the performance increase is significant.

As for the global analysis, the results remain really low which, as already explained, is most likely coming from the low level of information present in the dataset.

Another interesting thing to say is that, on the small dataset, as the detection is performed class by class, the detector is trained with only 25 samples which is a really low number of samples, the detector, however, manages to outperform the results from the global detection even in that configuration.

On the boxplot, the median is higher for most of the detectors, hierarchical clustering being on only one where no obvious benefit is visible. It's interesting to see that the "boxplot" method is the one who gained the most. This is easily explained by the fact that it performs a mono variate analysis for its detection. Removing the samples from the other class therefore allows it to more easily find the deviant samples. This increase is also visible on the results from the statistical test (figure 5.23) where this method comes in fourth position when doing the detection class by class while it was in seventh position before. The rest of the detectors ranking follow the usual pattern with, as always, hierarchical clustering at the end.

The boxplot per dataset's size variation from figure 5.24 are very similar to the ones from the global analysis, the only notable difference is the fact they tend to have a slightly reduced variation for the datasets with 200 features. This variance reduction is most likely not visible in smaller dataset size ($n = 50$) because of the variance increase caused by the really small dataset size when considering only one class and is not visible on the higher dimension due to its low performances.

6.4 Unsupervised

The results from the experiment on our synthetic datasets presented in section 5.4.2 are following the tendencies present in the two-dimensional tests performed in the Evaluation protocol chapter.

Removing the true or detected outliers both improves the classification performances more significantly when removing the outliers from both the training and testing sets.

In figure 5.20, you can see that the improvement offered by removing the outliers coming from the same distribution than the normal points is much more limited than for the two other kinds of perturbation. This was to be expected as, even if they are at the tail of their distribution, they are still consistent with it. The improvement offered by the outlier detection is still visible but unfortunately, the classification task being a bit too easy with that setup, the classifiers' performances are close to perfect which does not leave a lot of room for improvement.

The classification gain is much more visible in the other kind of perturbations such as mislabeled samples (the switch of class for some samples) or more extreme deviations which could have been caused by errors in the encoding or recording of the samples. In those setup, the gain is of a few percents.

Without surprise, almost all the statistical tests from figure 5.26 are showing that the classifiers working on the cleaned datasets have better results than the others in all the different kinds of perturbations with a small advantage for the complete removal of the outliers over the imperfect removal performed by the outlier detection.

Although the Friedman test performed on the external deviation (figure 5.25a) has a p.value of 0, indicating a difference between the classification performances, the confidence distance for the Nemenyi test is not big enough to clearly establish the difference. Looking at the scores and the average classification results, this test result is, however, easy to interpret. The test does not see any difference between predict and oracle as, due to the really high classification performances, their accuracy is of almost 100% on all the tests. On some datasets, all the classifiers have a perfect score which increases the number of ties between them and therefore makes it more difficult to differentiate the performances of the classifiers.

REED we will close our experiments with a quick look at the impact of our outlier removal on the classification performances in a real setup using REED. Of course, as those datasets are completely unsupervised, we don't know which points are actual outliers and cannot compare the performance of our outlier detection with a ground truth. Some of those datasets might not contain any true outlier, in which case, no significant improvement should be expected from removing samples.

Looking at table 5.21, we can see that , on average, removing the predicted outliers slightly increases the classification performances while removing random points does not positively impact the results.

The high performance gains from removing the samples that have been identified as less likely to be outliers by our outlier detection is irrelevant in the context of outlier detection but displays an interesting behavior that we thought worth explaining as we were first naively thinking that they would have a really bad impact on the results and were really surprised to see the opposite behavior. The outlier detection for those experiments have been performed globally, not class by class. As the points which are the most difficult to classify are the points located around the decision boundary, they are the points which are between the two classes. When performing a global outlier detection, if there is an overlap in the two classes, those points will be seen as having a normal behavior as they can be associated with the behavior of both classes. From the outlier detection point of view, those points therefore have a really low score, which explains why removing the samples with the lowest outlier scores has such a big impact on the performances of the classification results.

There is still an interesting conclusion that we can draw from this behavior: one of the concerns of DNAlytics is to cause an incorrectly high accuracy in REED and not to be able to reproduce those results with a manual analysis. They therefore are worried that the outliers detection would remove "tricky points" hard to classify but ultimately not incorrect. Those results tend to show that , when performing a global outlier analysis, unless the class separation

is really good leading to a really sparse region around the boundaries, the outlier detection will not target those "tricky" points. They might remove those points only if the boundary region is really sparse compared to the rest of the space. But in this case, the classification performances would not be highly impacted as the impact of those points would be negligible.

6.5 Threshold discussion - Visualization of the scores distributions

In the different experiments, we always used a pre-defined fixed threshold for determining which points were considered or not as outliers, but, as already mentioned in 2.1.3, it is common for the outlier detector to return scores which follows a characteristic distribution.

When the detectors confidently detect outliers, most of them give a high score for the detected samples and a really low score for the inliers.

The boxplot presented in 5.27 have been selected as they present interesting behaviors:

one class SVM in this case, does not follow this typical distribution as most of the points are near 0.005. As one class SVM is basing its score on the distance between the points to the decision boundary, this behavior means that there is either no points detected as outliers by the algorithm (all the points are on the same side of the decision boundary) or that there is almost no margin separating the normal points from the outliers;

Two phases clustering the shape of the score distribution returned by Two phases clustering is really interesting and is caused by the fact that the scores are computed by batches, giving the same score to several points;

PCA + LOF for this detector, the behavior exactly follows the expected one as the distinction between the detected outliers and the inliers is clearly visible.

Ensemble once again, the behavior is following the expected one with all the inliers grouped at the bottom of the scores and the outliers sticking out. The interesting thing here is that, as the scores are composed of different detector, the confidence is more visible as the points detected by many detectors are getting a score significantly higher than the others.

Looking at the scores, we can clearly see that the AUC is not reflecting the entire situation as it misses this notion of confidence.

Those results, all displaying good detection performances as the outliers are almost always detected first, are showing that fixing a predefined threshold, for example, based on the standard deviation, is not that easy as all the distributions are different.

Of course, in this case, we know that the dataset contains 5% of outliers so determining how many samples need to be removed is easy but in a real dataset, in which we don't know the real number of outliers, this might cause non-outliers to be removed as well.

6.6 Analysis conclusion

6.6.1 Base detectors

Some tendencies emerge from the divers experiments we performed:

- PCA + LOF is performing really well on the simulated datasets but, as already mentioned, the fact that all those datasets have been generated using the same generative process should be taken into consideration;

- LOCI, however, performs really well on most of the datasets and, as it returns the same scores of every sample when it fails to detect the outliers, it is well suited to be integrated in an ensemble;
- Even if it performs correctly on the public datasets, hierarchical clustering is clearly less efficient than the other which was to be expected, seeing the flaws exposed in the section 2.3 about the curse of dimensionality as it builds its tree randomly if the difference in distances are too small and therefore could give high scores to normal points.

6.6.2 Ensemble

Even if the ensemble is never the best method, it consistently returns results which are among the bests even though it includes hierarchical clustering which is always significantly worst than the others, confirming the stabilizing factor offered by the ensemble.

The improvement offered by the bagging is also clearly sensible and allows to reduce the impact of the curse of dimensionality.

Chapter 7

Conclusion

The different experiments performed during this work have all been based on the notion that an outlier is a sample that is less likely to be drawn from the same distribution than the normal points, whether using public datasets (considering the minority class as outliers) or using simulated datasets (with the last percentile of the distribution as outliers).

Even if the model of "outliers" present in the two kinds of datasets is really different, Local Correlation Integral consistently returns results which are among the best and when used together with an ensemble method, it does not degrade the results when it fails to detect the outliers which makes it a really good candidate. Even though it suffers in high dimensionality and is computationally more expensive.

Local Outlier Factor which is considered as a reference in the literature is much lighter to run and returned the best results in most of the case.

The ensemble method, even if not returning results as good as the best detectors confirms it stabilizing factor which tends to suggest that using ensemble is a really good solution in an unsupervised environment requiring robustness such as REED.

Moreover, the improvement offered by the bagging seems really promising, as expected due to the strong impact of the curse of dimensionality. It would be really interesting to perform a comparison of different bagging approach as we only tested the most basic one and already noticed a sensible performance increase .

Unfortunately, most of our experiments are based on synthetic datasets due to the lack of good unsupervised assessment methods. The literature does not offer any real leads on how to tackle the lack of ground truth, preventing to draw any strong conclusions from the unsupervised experiments.

Had we had more time...

This work was really interesting but unfortunately, the problem being really vast, there are still many things that ought to be investigated. Had we had more time, we would have gone more in detail in the direction of the dimensionality reduction, as it clearly has a determining impact.

Other leads for additional work might be:

- increase the scope of the base detectors as we only considered a small subset of the huge number of available algorithms;
- investigate other feature selection methods, such as *High Contrast Subspaces for Density-Based Outlier Ranking* [19], as Mutual Information seemed really promising but is computationally too expensive in high dimensional setup;
- investigate other ensemble improvements such as the automated selection of base detectors, reduce the impact of the inliers, improve the score composition;
- the automated detection of the optimal threshold on the outlier scores based on the shapes of the score distributions.

Appendices

Appendix A

Additional figures

A.1 Preprocessing by method for public datasets

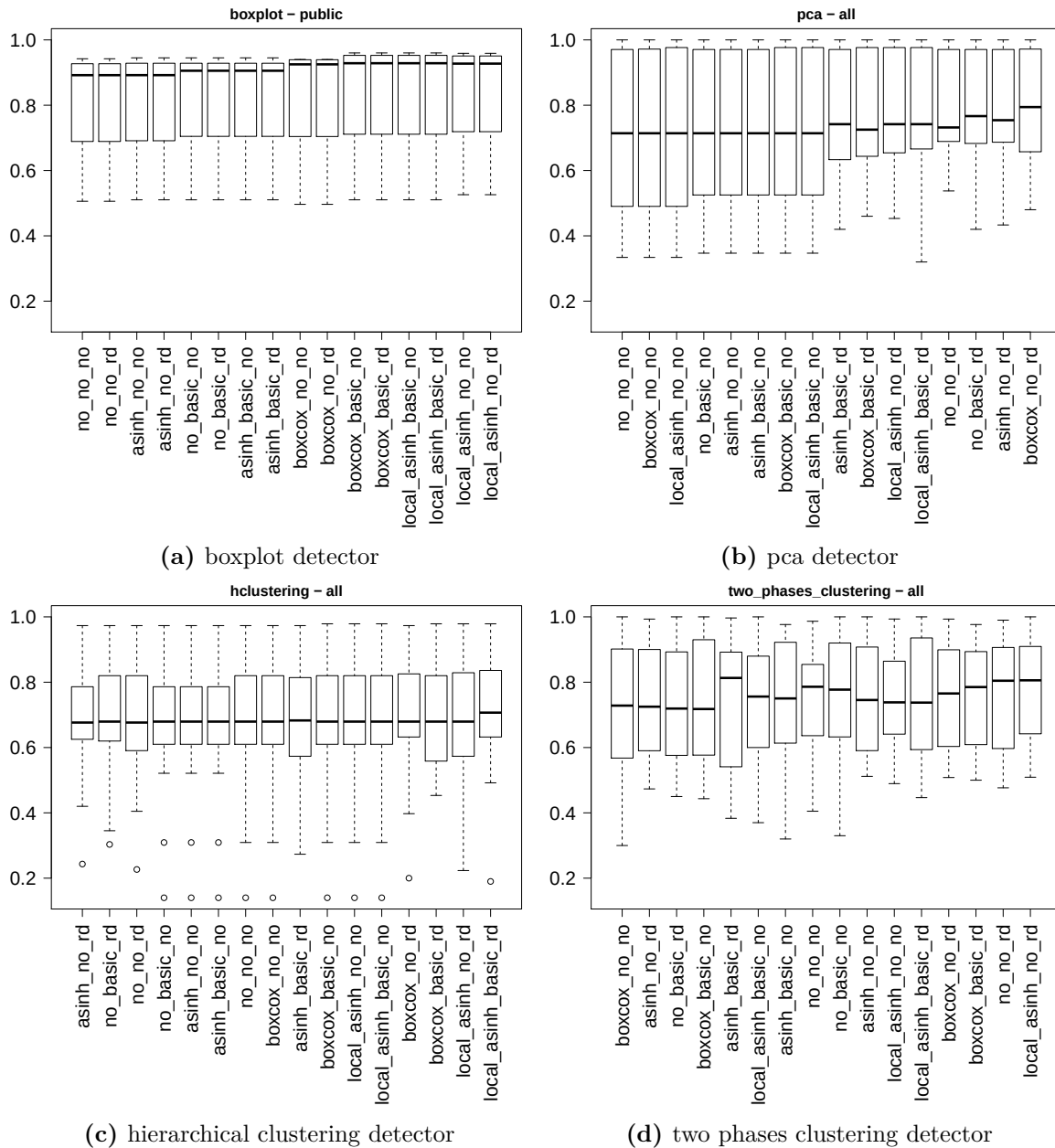
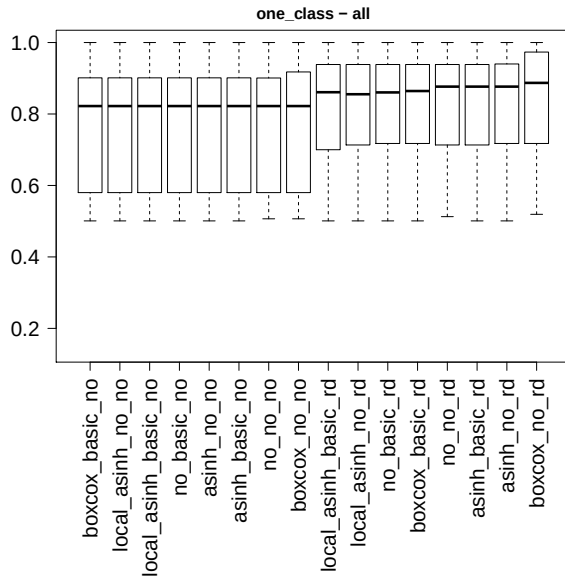
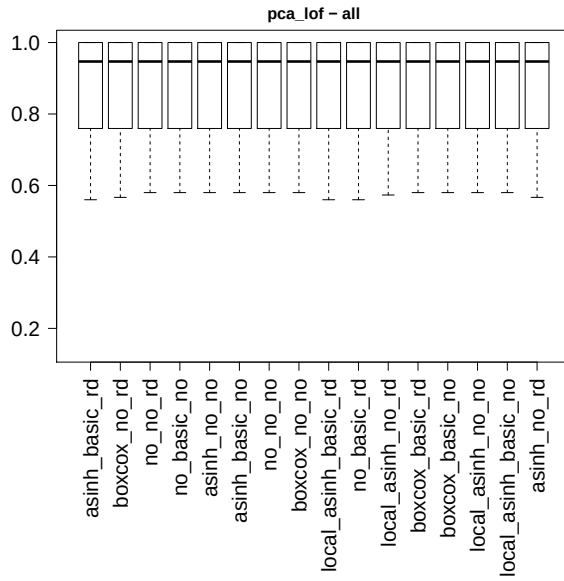


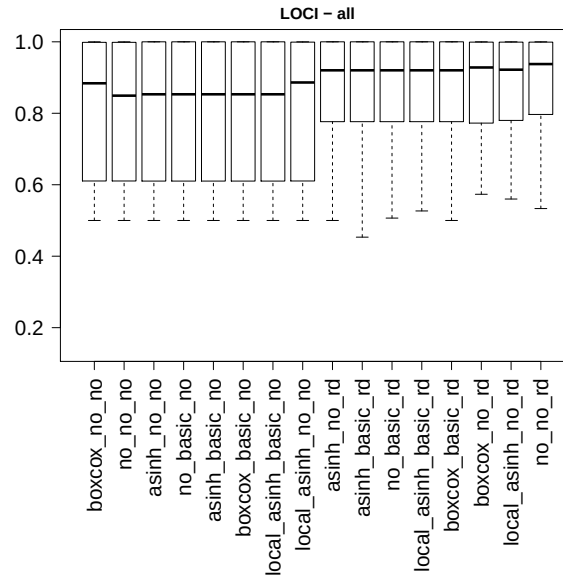
Figure A.1: Impact of the different pre-processing configurations on the base detector for the all files



(e) one class SVM detector



(f) pca + lof detector



(g) LOCI detector

Figure A.2: Impact of the different pre-processing configurations on the base detector for the all files

A.2 Boxplot visualisation of the scores

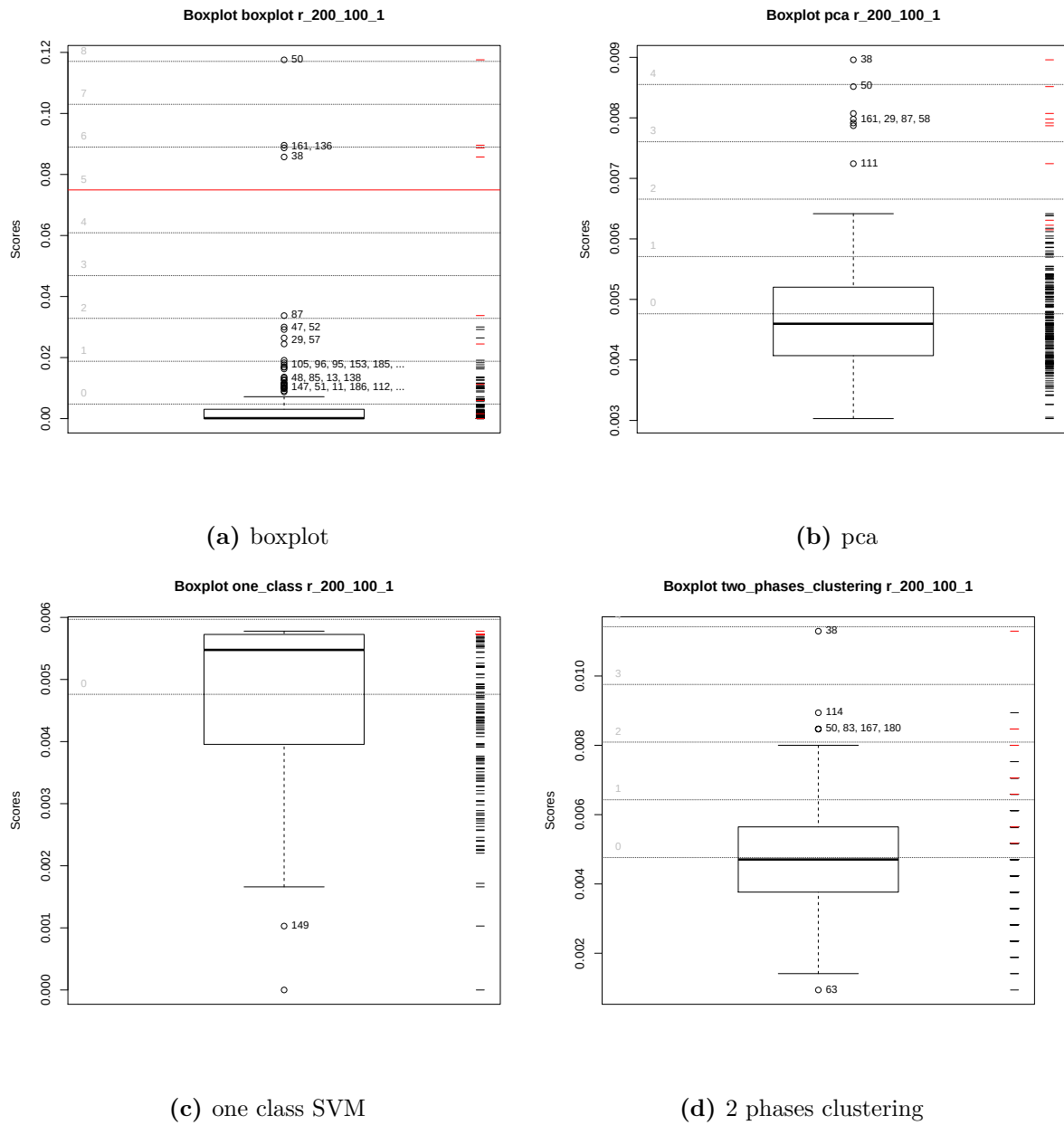
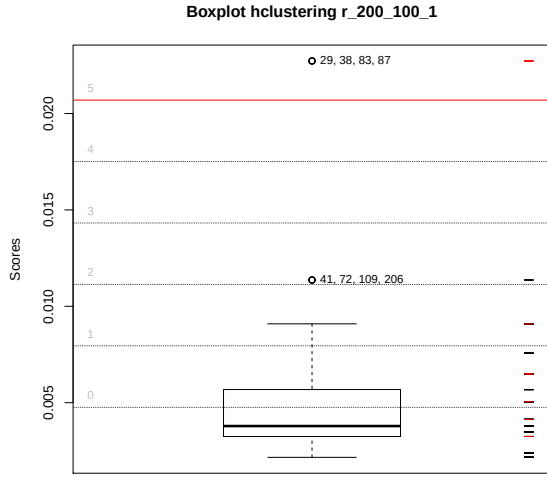
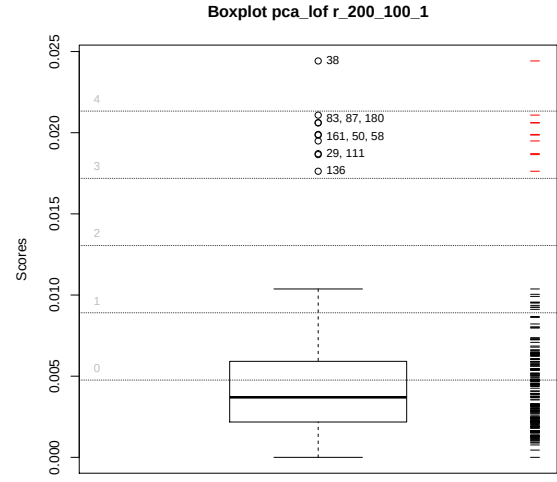


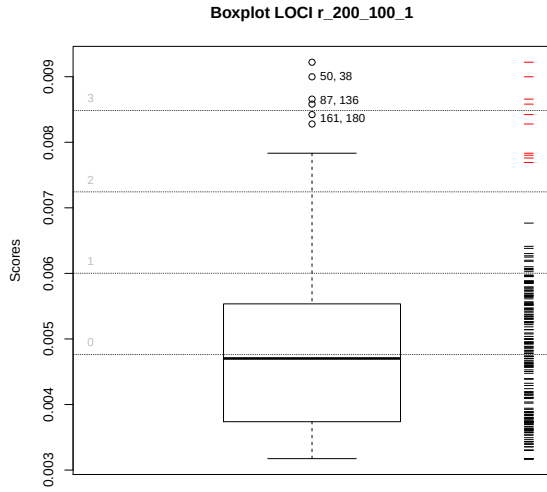
Figure A.3: Boxplot visualisation of the scores returned by the different detectors



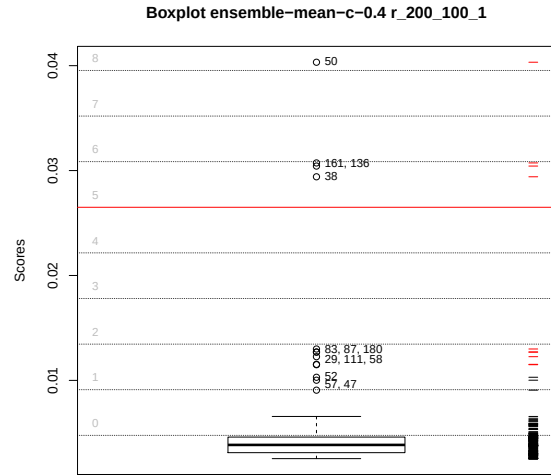
(e) hierarchical clustering



(f) pca lof



(g) LOCI



(h) ensemble

Figure A.4: Boxplot visualisation of the scores returned by the different detectors

Bibliographic references

- [1] Charu C. Aggarwal. *Outlier Analysis, Second Edition*. Springer, 2016.
- [2] Charu C Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, 17(1):24–47, 2015.
- [3] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.
- [4] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15. ACM, 2013.
- [5] V Barnett and T Lewis. Outliers in statistical data: Chichester. *John Wiley&Sons*, 1994.
- [6] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.
- [7] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.
- [8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.
- [9] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.
- [10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.
- [11] Timothy De Vries, Sanjay Chawla, and Michael E Houle. Finding local anomalies in very high dimensional space. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 128–137. IEEE, 2010.
- [12] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.
- [13] H Jair Escalante. A comparison of outlier detection algorithms for machine learning. In *Proceedings of the International Conference on Communications in Computing*, pages 228–237, 2005.
- [14] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
- [15] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Data Mining, 2006. ICDM’06. Sixth International Conference on*, pages 212–221. IEEE, 2006.

- [16] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.
- [17] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.
- [18] Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. Two-phase clustering process for outliers detection. *Pattern recognition letters*, 22(6):691–700, 2001.
- [19] Fabian Keller, Emmanuel Muller, and Klemens Bohm. Hics: High contrast subspaces for density-based outlier ranking. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 1037–1048. IEEE, 2012.
- [20] Marius Kloft and BERLIN DE. Hidden markov anomaly detection. 2015.
- [21] Antonio Loureiro, Luis Torgo, and Carlos Soares. Outlier detection using clustering methods: a data cleaning application. In *Proceedings of KDDNet Symposium on Knowledge-based systems for the Public Sector*, 2004.
- [22] Stijn Luca, David A Clifton, and Bart Vanrumste. One-class classification of point patterns of extremes. *Journal of Machine Learning Research*, 17(accepted), 2016.
- [23] Annette M Molinaro, Richard Simon, and Ruth M Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.
- [24] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE, 2003.
- [25] Jérôme Paul. *Feature Selection from Heterogeneous Biomedical Data*. PhD thesis, Université catholique de Louvain, 2015.
- [26] Yaling Pei and Osmar Zaiane. A synthetic data generator for clustering and outlier analysis. 2006.
- [27] Shebuti Rayana. Odds library, 2016.
- [28] Karanjit Singh and Shuchita Upadhyaya. Outlier detection: applications and techniques. *International Journal of Computer Science Issues*, 9(1):307–323, 2012.
- [29] Hadley Wickham. *Advanced R*. CRC Press, 2014.
- [30] Huan Xu, Constantine Caramanis, and Shie Mannor. Outlier-robust pca: the high-dimensional case. *IEEE transactions on information theory*, 59(1):546–572, 2013.
- [31] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.
- [32] Arthur Zimek, Ricardo JGB Campello, and Jörg Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *Acm Sigkdd Explorations Newsletter*, 15(1):11–22, 2014.

