POLITECNICO DI TORINO

Master Course
in Mathematical Engineering

Master Thesis

Independent and Sequential Ensemble Methods for Anomaly Detection

Supervisor
Prof. Gianluca MASTRANTONIO

Company tutor
Ing. Francesco SARACCO

Candidate
Maria Sebastiana DI BLASI

Accademic Year 2022-2023

To my family,
anchor of my life!

Summary

Anomaly detection problems are particularly important in various real-word contest such as fraud detection, finance, intrusion detection and cyber-security. Several methods that are presented for outlier detection work well in peculiar fields but fail if they do not meet some characteristics. We develop algorithms that can be applied in a several areas and can help in solving many real problems.

In this work we focus on ensemble methods for anomaly detection in static dataset to show that combining different base learners we achieve better performances than most of the base algorithms. Before analyzing the different combinations, i.e. the ensemble strategies, we present some base algorithms being the basic components of such methods. We present and analyze different ways to combine the different base learners such as the score averaging method, the maximum score combination, the averaging ranking approach and majority voting. The ensemble methods are classified in independent and sequential. In the independent ensemble methods, the base learners, that are assumed to be independent of each other, are applied to the entire dataset and the obtained scores are combined using one of the methods mentioned above. In the sequential ensemble methods the base learners are applied sequentially and every other data in the base learner is having some dependency on previous data. We report and discuss the results obtained from the implemented ensemble methods and we compare them with those obtained using basic anomaly detection algorithms. The best performances are obtained by combining different algorithms. Some of the base learners used are present in the Python libraries, others have been implemented by us.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of outlier detection techniques is to find the objects of the dataset that are meaningfully diverse from other observations because they are from different distributions.

Anomaly detection problems emerge in multiple applications such as security, financial fraud, medical failure, cyber-security and intrusion detection and they are studied more and more by more researchers being very important in many fields. The outliers are considered abnormal respect to other observations in the dataset because they are characterized by patterns that haven't a well-defined normal behavior. Since the "normal" behavior of observations is not static but changes over time it is really hard to identify it in the real-world applications.

Many researchers have formulated many anomaly detection methods based on statistics, machine learning and theory techniques. However, each algorithm detect a single kind of anomaly because is developed for specific class of problem, but in real applications are present different types of anomalies in the dataset and the goal is to be able to identify as many anomalies as possible and therefore also of different types.

Ensemble methods were developed to achieve better performances and more robust solutions because they combine different algorithms. The diversity is really important for the choice of algorithms because it permit to correct the error of the previous algorithm. In this work we will present different algorithms and approaches used for anomaly detection.

In particular, in the first chapter we report the most commonly used definitions to identify an anomaly. In the second chapter we describe some base learners that are very important in anomaly detection and that, in general, can be characterized as distance-based, density-based and rank-based. In the third chapter we expose in detail the independent and sequential ensemble methods and the various combinations that can be used to build these methods. In the last chapter we expose and define the metrics often used in outliers detection and also in our experiments to compare the different methods and

choose the best one. We report a small description of the datasets used for our experiments, finally we show and compare the obtained results with the base learners and with ensemble methods, both independent and sequential.

Thus, the goal of this work is to show that ensemble methods allow detecting anomalies of a dataset with higher accuracy than basic learners.

Chapter 2

Anomalies

Anomalies or outliers are substantial variations from the norm, i.e observations that have a different pattern than the normal one. Many definitions of anomaly have been presented in the literature and the most commonly used are the following:

- "An outlier is an observation that deviates so much from other observations as to arouse suspicions that is was generated by a different mechanism" (9).
- "Anomalies are patterns in data that do not conform to a well defined notion of normal behavior" (5).
- "An outlier is an observation that lies outside the overall pattern of a distribution" (17).
- "An outlier in a set of data is an observation or a point that is considerably dissimilar or inconsistent with the remainder of the data" (20).

To describe the normal behavior of the data it is necessary that the observations satisfy some assumptions. The first is the stationarity, i.e., the data must be generated by an underlying process that doesn't change significantly over time. Anomaly detection models are built using predictions from past data, therefore the stationary condition is verified if the statistics that characterized a system in the past continue to characterize the system in the future.

There are many outlier problems in real-world applications, but we focus on "**Point outliers**" for static dataset, i.e., we want to detect individual observations that are distinct with respect to the rest of dataset. For example, in Figure (2.1), data points A, B, C and D are indicated as anomalies since they are not in any cluster and they are far away

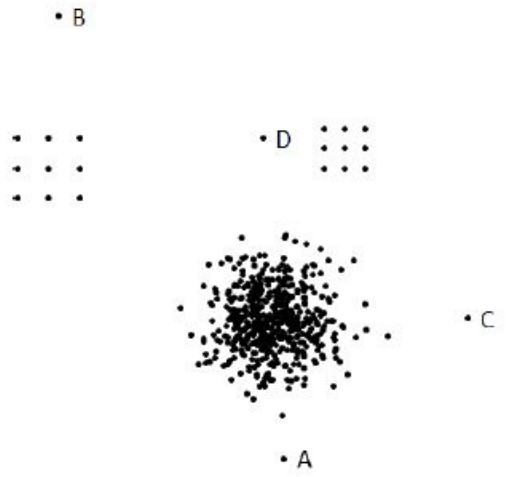

Figure 2.1. **Outliers in a static dataset:** Data point A, B, C and D are anomalies with respect to the rest of the observations in this two dimensional dataset.

from the majority of the observations. So, using the "neighborhood" approach, they are considered different from the rest of dataset.

Point outliers in the static dataset are usually characterized by abnormal attributes with respect to the rest of dataset. More formally, let D a given dataset and p an observation such that $p \in D$, we want to find the set $O(p)$ $\forall p$, i.e the set of outlierness of p, and $O_{threshold}$, i.e the set of thresholds, such that if $O(p) \geq O_{threshold}$, then p is an outlier, otherwise it is considered inlier.

Unlike supervised data mining techniques, outlier detection is typically an unsupervised learning problem because the behaviors or patterns of outliers are unknown. Unsupervised algorithms aim at partitioning the dataset in two classes: the expected data points and the anomalous ones. The anomalies are rare events, and therefore most observations in the dataset exhibit "normal behavior". In the next chapter are presented the main principal algorithms used for anomaly detection. When an anomaly detection approach is used, three rates have to be considered:

- *Correct Detection:* detected anomalies correspond to the true anomalies in the dataset.
- *False Positives:* some anomalies in the dataset are classified as normal but actually they are abnormal.
- *False Negatives:* some normal observations are classified as outliers but actually they

are inliers.

In real life it is impossible to be able to detect 100% of the anomalies, i.e. to obtain an anomaly detection algorithm characterized by a "Correct Detection rate" equal to 1. In fact, the goal is to be able to identify the largest number of observations, inliers and outliers, correctly.

In the last chapter, the presented rates representing the three characteristics will be discussed in more detail.

Chapter 3

Anomaly Detection

This section present the main traditional approaches to anomaly detection that we apply in ensemble models. They can be classified as *distance-based*, *density-based* and *rank-based*. In the distance-based methods the points are considered more anomalous the more they are far from others. In the density-based methods the points that are considered more anomalous are those with relatively low density. In the rank-based methods, an anomalous point can be defined as a point whose closest neighbors do not have that point as one of their closest neighbors.

3.1 Distance Based Anomaly Detection Approaches

In this section we analyze anomaly detection approaches based on distance measures. To simplify the discussion we assume that the observations belong to continuous space but the methods also applies to discrete and categorical attributes with a suitable distance metric. In continuous space a data point is "different" from others if its distance to other observations is large. Distance-based algorithms differ from the choice of the type of distance. We present some of the most common distances.

- **Distance to All Points**

  Let D be a dataset, we compute the distance for each point $p \in D$ against all points in D. The sum of distances from all points can be used as the anomalousness metric, i.e.,

  $$\alpha(p) = \sum_{q \in D} d(p, q).$$

  The most anomalous point is farthest from all points in the dataset.

- **Distance to Nearest Neighbor**

  Let D a dataset, we calculate for each point $p \in D$ the distance to the nearest point in the dataset. This distance can be used as the anomalousness metric, i.e.,

  $$\alpha(p) = \min_{q \in D, q \neq p} d(p, q).$$

  The most anomalous point is the one farthest from its nearest neighbor.

- **Average Distance to K Nearest Neighbors**

  Let D a dataset and $N = |D|$ the number of points in the dataset. Let $K < N$ a parameter that identifies the number of nearest neighbors that are considered. For each point we compute the average distance to the K nearest neighbors that can be used as anomalousness metric, i.e.,

  $$\alpha(p) = \sum_{j=1}^{K} \frac{d(p, Near(p, j))}{K},$$

  where $Near(p, j)$ indicates the j-th nearest neighbor of point $p \in D$. The most anomalous point is the one with the greatest average distance. Instead of the average distances can be used the sum of distance as indicator of the anomalousness.

- **Median Distance to K Nearest Neighbors**

  In the average distance is used the arithmetic average but it is not very robust. For example if one or more points are added, the outcome can change drastically. A way to solve this problem is used the median that is a more robust measure because it is less sensitive to noise in the data although it requires more calculations.

We present some of the most common distance-based algorithms used in anomaly detection: "*Distanced Based Outlier*", "*Local Correlation Integral*", "*Resolution-Based Outlier*" and "*Nearest Neighbor*".

3.1.1 Distance Based-Outlier Algorithm

This algorithm uses the definition of distance based (DB) outliers, *DB-outliers*, given by Knorr and Ng in(12), that it is the following: "an object p in a dataset D is a $DB(\pi, R)$-outlier if at least a fraction πD (where $0 \leq \pi \leq 1$) of the objects in D are at a distance greater than r from p". The parameter r and π are set by user. The set $N_p(r)$ contains the neighbors of p which are located at a distance less than or equal to r, i.e.,

$$N_p(r) = \{q : q \in D \quad and \quad d(p, q) \leq r\}.$$

Algorithm DB-outlier
Require: π, r, D
Ensure: List of outliers

```
1: O = ∅
2: for p ∈ D do
3:     N_p(r) = NULL
4:     for q ∈ D
5:         if dist(p, q) ≤ r then
6:             Insert q in N_p(r)
7:         end if
8:     end for
9:     if |N_p(r)| ≤ (1 − π)|D| then
10:        Insert p into O
11:    end if
12: end for
```

Figure 3.1. **DB-Outlier Algorithm**

$N_p(r)$ is called *sampling neighborhood* of p. The point p is considered outlier if

$$|N_p(r)| =\leq (1-\pi)|D|,$$

where $|D|$ is the cardinality of the dataset, i.e., the number of points in the dataset and $|N_p(r)|$ is the number of points in $N_p(r)$. In the Figure (3.1) is presented the pseudo-code of the DB-outlier algorithm.

This algorithm is not efficient because to determine all anomalous observations is time consuming. A more efficient alternative is the *Index-Based Algorithm*. Let N be the number of objects in a dataset D and we consider $N_p(r)$ defined as above, i.e., the set of objects in D with distance r from p. Let π be the minimum fraction of objects in D that must be outside the r-neighbourhood of an outlier. Let K the maximum number of objects within the r-neighbourhood of an outlier i.e., $K = N(1-\pi)$. Therefore, the problem of finding all $DB(\pi, r)$-outliers can be solved by counting the number of points within a distance r from the point p. A soon as $(K+1)$ neighbours are found in the r-neighbourhood, the search stops, and p is declared a non-outlier, otherwise p is an outlier.

An other variation of DB-outlier algorithm is *Nested-Loop Algorithm*. This method find all neighbours of object p within distance r using a nested loop approach, i.e.: for each object $p \in D$, it is sequentially computed the distance between p and an object $q \in D$ until K neighbours with distance r have been found. If K neighbours within distance r from p have been found then p is not outlier, otherwise p is an outlier. Another approach is *Cell-Based Algorithm* where the whole space is partitioned into small multi-dimensional

cells. Each object $p \in D$ is associated to a specific cell. In order to decided if an object p is an outlier or an inlier it considers only the objects in the same cell of p or in p's neighbour cells. By using some specific choices of cell sizes related to parameter r this approach can be more efficient. For example, with a dataset D with dimensionality d a good choice is to partition the whole space into cells of length $l = \frac{r}{2\sqrt{d}}$.

3.1.2 Local Correlation Integral (LOCI) Algorithm

Local Correlation Integral is a multi-point approach. We define a new set $N_p(\alpha r)$, called *counting neighborhood* of p. It is the set of points with distance αr from p, where the parameter $0 < \alpha < 1$ is predetermined. We indicate with $n(p, \alpha r)$ the number of points in $N_p(\alpha r)$ and with $\hat{n}(p, r, \alpha)$ the average of $n(q, \alpha r), \forall q \in N_p(r)$, i.e.,

$$\hat{n}(p, r, \alpha) = \frac{1}{|N_p(r)|} \sum_{q \in N_p(r)} n(q, \alpha r),$$

Given r and α, the *Multi-granularity Deviation Factor (MDEF)* at point $p \in D$ is defined in the following way:

$$MDEF(p, r, \alpha) = 1 - \frac{n(p, \alpha r)}{\hat{n}(p, r, \alpha)}.$$

MDEF can be on both positive and negative. If MDEF is negative, then p is not anomalous, whereas a high positive value indicates that p is more likely to be an outlier because p has relatively few near-neighbors, when it is compared to other points in the same region. Local Correlation Integral Algorithm to determine the outlierness of the point p proceeds in the following way:

- The value of r_{max} is set to $\approx \alpha^{-1} \max_{p,q \in D} \delta(p, q)$, while r_{min} is chosen so that in the meaningful neighborhood there are approximately 20 points. $\delta(p, q)$ is the distance from p to q and $r \in [r_{min}, r_{max}]$.
- $MDEF(p, r, \alpha)$ is computed $\forall r \in [r_{min}, r_{max}]$.
- The deviation standard of $MDEF(p, r, \alpha)$ is computed:

  $$\sigma_{MDEF}(p, r, \alpha) = \frac{\sigma_{\hat{n}}(p, r, \alpha)}{\hat{n}(p, r, \alpha)},$$

  where $\sigma_{\hat{n}}(p, r, \alpha)$ is the deviation standard of $n(q, \alpha r)$.
- An observation p is considered outlier if $\forall r \in [r_{min}, r_{max}]$ its $MDEF$ is large, more formally if $MDEF(p, r, \alpha) > k_\sigma \times \sigma_{MDEF}(p, r, \alpha)$, then p is indicated to be an outlier.

In the Figure (3.2) is presented the pseudo-code of Local Correlation Integral Algorithm with $\alpha = \frac{1}{2}$ and $k_\sigma = 3$.

Algorithm Exact LOCI algorithm

1: **for** each $p \in D$ **do**
2: Find $N_p(r_{max})$;
3: Compute $\delta(p, p_{m-NN})$ and $\delta(p, \alpha p_{m-NN})$ for $1 \leq m \leq N$, where p_{m-NN} denotes the m^{th} nearest neighbor of p;
4: Sort the list of these δs in ascending order of magnitude;
5: For each r, in the sorted list, calculate $n(p, \alpha r)$ and $\hat{n}(p, \alpha r)$;
6: Compute $\mathbf{MDEF}(p, \alpha)$ and $\sigma_{\mathbf{MDEF}}(p, \alpha, r)$;
7: if $\mathbf{MDEF}(p, \alpha, r) > 3\sigma_{\mathbf{MDEF}}(p, \alpha, r)$, then flag p as a potential outlier.
8: **end for**

Figure 3.2. **Exact LOCI Algorithm**

3.1.3 Resolution-Based Outlier Detection Algorithm

An another method to measure the outlierness of an observation $p \in D$ is the *Resolution-Based Outlier Detection* that use different resolutions and then aggregate the results. All observations are isolated points, i.e. are outliers, when the resolution is highest, while all observation belong to one cluster when the resolution is the lowest and thus all points are inliers. The maximum resolution is obtained by considering $r_1 < \min_{i \neq j, p_i, p_j \in D} d(p_i, p_j)$, so $N_p(r_1)$ contains only one point p, $\forall p \in D$. The smallest resolution is obtained by considering $r^* = \max_{p,q \in D} d(p, q)$, so all observation in D belong to one cluster. In the Resolution-Based Outlier Detection we consider $p, q \in D$ and $r > 0$, q is a close neighbor of p if $d(p, q) \leq r$. If q is a close neighbor of p then it is a point of cluster. We proceed in this way for all points of the cluster until all close neighbors of observations in a cluster are contained in the cluster. Considering r_1 and r^* we can define the intermediate resolutions choosing $r_2 < r_3 < .... < r_R = r^*$. If we define the equal spacing as $\Delta_R = (r^+ - r_1)/R$ then $r_i = r_i - 1 + \Delta_R$ and the resolution-based outlier factor (**ROF**) is given by:

$$ROF(p) = \sum_{i=1}^{R} \frac{cluster - size(p, r_i - 1) - 1}{cluster - size(p, r_i)},$$

where $cluster - size(p, r_i)$ is the number of points in the cluster that contains p. The most anomalous point is that with the smallest ROF.

3.1.4 Nearest Neighbor Algorithm

In the previous approach the number of points within a fixed radius r were used to determine the outlierness of observations. In the *Nearest Neighbor Algorithm* is used an alternative that is the following: a point can be considered outlier if its neighbors are far away.

Algorithm k-NN outlier
Require: k, n, D
Ensure: O has n of outliers

```
1: O = ∅
2: for p ∈ D do
3:     N_p = NULL
4:     for q ∈ D
5:         if |N_p| < k then
6:             Add q in N_p
7:         else
8:             if max{dist(p,s)|s ∈ N_p} > dist(p,q) then
9:                 Add q in N_p and remove the first s in N_p such that
                   dist(p,s) > dist(p,q)
10:            end if
11:        end if
12:    end for
13:    NN(p,k) = max{dist(p,s)|s ∈ N_p}
14: end for
15: for p ∈ D do
16:    if |O| < k then
17:        Add p in D
18:    else
19:        if min{NN(s,k)|s ∈ O} < NN(p,k) then
20:            Add p in O and remove the first s in O if NN(s,k) < NN(p,k)
21:        end if
22:    end if
23: end for
24: return O
```

Figure 3.3. **K-NN Algorithm**

The outlierness of point p can be calculated using the distance of its neighbors. Let $NN(p,k)$ be the k-th nearest neighbor of p and we compute the distance from p to the k-th nearest neighbor, i.e., $d(p, NN(p,k)) = d_k(p)$. If $d_k(p)$ is large then p is considered anomalous. The points with the highest values of $d_k(p)$ are indicated as outliers. In the Figure (3.3) is presented the pseudo-code of K-Nearest Neighbor Algorithm.

3.2 Density Based Anomaly Detection Approaches

In the density based algorithms the idea is to look at the "local" density of a observation with respect to the density of its neighbors. The neighborhood is built by considering k nearest neighbors, where k is an integer number. In the density based approaches if the density at a point is smaller than the density of its neighbors, then the point is

anomaly. The main difference with the distance-based algorithms is in the definition of the local behavior and in the related density. The local density of a point p is defined as the reciprocal of the average distance among the k nearest points to p and the outlier score of p varies inversely with the local density at p. In this way we can compute the outlier score of each point and sort the scores in decreasing order. The points such that the anomaly scores are greater than a pre-defined threshold are considered outliers. The density based anomaly detection approaches assume a symmetric distance function, i.e., $dist(x, y) = dist(y, x) \quad \forall x, y \in D$. Let $N_k(x)$ be the set of k nearest neighbors of a observation $x \in D$, we consider the k-distance $d_k(x)$ defined as in the Nearest Neighbor Algorithm, then we have that $N_k(x)$ is defined as:

$$N_k(x) = \{y \in D \setminus \{x\} \quad | \quad dist(x, y) \leq d_k(x)\}.$$

The reverse nearest neighborhood of a point $x \in D$, $R_k(x)$, is the set of points $y \in D$ such that x is among the k nearest neighbors of y and it is defined as:

$$R_k(x) = \{y \in D \quad | \quad x \in N_k(y).\}$$

$R_k(x)$ can be empty because x can not be in any of the k nearest neighbors sets of its k nearest neighbors. We present some of the most common distance-based algorithms used in anomaly detection:"*Local Outlier Factor Algorithm*", "*Connectivity-Based Outlier Factor Algorithm*" and "*INFluential Measure of Outlierness by Symmetric Relationship*".

3.2.1 Local Outlier Factor (LOF) Algorithm

Local outlier factor (LOF) measures the degree of anomaly of a point x using its local neighbors. For each point $x \in D$ is computed its LOF and x is an outlier if the value of its LOF is large. To calculate the LOF score it is possible proceed as follow:

- Find the distance $d_k(p)$, i.e., the distance between p and its k-th nearest neighbors. The most used distance is the Euclidean distance, but any measure can be considered. Calculate $N_k(x)$, i.e.,the k nearest neighborhood set of each point $x \in D$.

- Compute the local reachability density of each point $x \in D$ that is defined as:

$$lrd_k(x) = \left[\frac{\sum_{y \in N_k(x)} reach - dist_k(x, y)}{|N_k(x)|}\right]^{-1},$$

  where $reach - dist_k(x, y)$ is the reachability distance of a point y from x and it is given by $reach - dist_k(x, y) = max\{d_k(y), d(x, y)\}$.

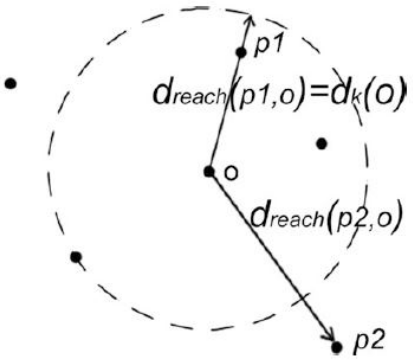

Figure 3.4. **Reachability distance:** $d_{reach}(p1, o)$ and $d_{reach}(p2, o)$ with k=3

- Compute the LOF score for each point $x \in D$:

$$LOF_k(x) = \frac{\sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)}}{|N_k(x)|}.$$

- The points are sorted in decreasing order of LOF. If the LOF values are large then the corresponding objects are outliers. In particular, if LOF score of a point is > 1 then the point is potentially an outlier, while if the LOF score is ≤ 1 then the point is an inlier.

In the Figure (3.5) is presented the pseudo-code of Local Outlier Factor Algorithm.

3.2.2 Connectivity-based Outlier Factor (COF) Algorithm

To solve one of the deficiencies found in the LOF, i.e., to detect outliers when in the dataset there are clusters of different shapes, it was proposed the *Connectivity-based Outlier Factor* (COF) that use a new method to compute the density that is the "connectivity", i.e., how an object connects to its neighbors, and use relative isolation to indicate whether an point is deviating from others. Given k, to compute the COF score for each point $x \in D$, we can proceed as follows:

- Define the distance between two non empty, disjoint set A e B as

$$set - dist(A, B) = min\{dist(x, y) \quad | \quad x \in A, y \in B\}.$$

- Let $N_k(x)$ be the set of k nearest neighbors of x and define the *set-based path (**SBN**)* of length k as the path starting from $x_1 :< x_1, x_2, ..., x_k >$ such that x_{i+1} is the

Algorithm LOF (Local Outlier Factor Computation)

Require: k, D

Ensure: L_k - LOF score for each object in D

1: $L_k = \emptyset$
2: **for** $p \in D$ **do**
3: $N_k(p) = NULL$
4: **for** $q \in D$
5: **if** $|N_k(p)| < k$ **then**
6: Add q in $N_k(p)$
7: **else**
8: Let $s^* \in N_k(p)$ be such that $dist(p, s^*) \geq dist(p, s)$ for all $s \in N_k(p)$;
9: **if** $dist(p, s^*) > dist(p, q)$ **then**
10: Replace $s^* \in N_k(p)$ by q
11: **end if**
12: **end if**
13: **end for**
14: $d_k(p) = max\{dist(p, s) | s \in N_k(p)\}$
15: **end for**
16: **for** $p \in D$ **do**
17: **for** $q \in D$ **do**
18: $d_{reach}(p, q) = max\{d_k(p), d(p, q)\}$
19: **end for**
20: **end for**
21: **for** $p \in D$ **do**
22: $l_k(p) = \frac{|N_k(p)|}{\sum_{q \in N_k(p)} d_{reach}(p,q)}$
23: **end for**
24: **for** $p \in D$ **do**
25: $L_k(p) = [\frac{\sum_{o \in N_k(p)} \frac{l_k(o)}{l_k(p)}}{|N_k(p)|}]$
26: **end for**
27: **return** L_k

Figure 3.5. **LOF Algorithm**

nearest neighbor of set $\{x_1, x_2, ..., x_i\}$ in the set $\{x_{i+1}, ..., x_k\}$ for $1 \leq i \leq k-1$. The SBN-path is generated starting from a single point x and in each iteration the nearest neighbor point is added to $SBN(x)$. The SBN is an ordered list of all neighbors of x.

- The *set-based trail (**SBT**)* is an ordered collection of $k-1$ edges with respect to a SBN-path $p =< x_1, x_2, ..., x_k >$. It is defined as a sequence $< e_1, e_2, .., e_{k-1} >$ such that the i-th edge e_i connects a point $o \in \{x_1, .., x_i\}$ to x_{i+1} and it is of the minimum distance. $|e_i| = set - dist(x_1, .., x_i, x_{i+1}, .., x_k)$ is called the *cost description* of edge e_i. In the Figure 3.6 is illustrated a SBT and a SBN. The SBN and the SBT define

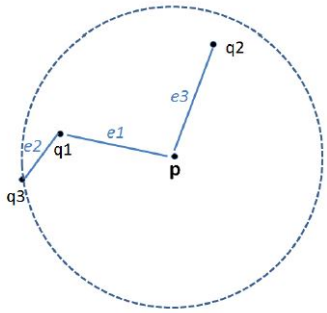

Figure 3.6. For k=4, the **SBN**-path of $p \in D$ is $\{p, q_1, q_2, q_3\}$ and **SBT** is $< e_1, e_2, e_3 >$.

the connectivity.

- Compute the average-chaining distance ($\mathcal{A}$) from x to $N_k(x)$ which is the weighted sum of the lengths of the edges, i.e.,

$$\mathcal{A}_{N_k(x)}(x) = \sum_{i=1}^{k-1} w_i \times e_i,$$

where w_i is the weight for edge e_i and it is proportional to the order in which is added to SBT set. In particular, the weight w_i is given by:

$$w_i = \frac{2(k-i)}{k(k-1)}.$$

- Compute the connectivity-based outlier factor (COF) for each point $x \in D$ as:

$$COF_k(x) = \frac{|N_k(x)| \cdot \mathcal{A}_{N_k(x)}(x)}{\sum_{y \in N_k(x)} \mathcal{A}_{N_k(y)}(y)}.$$

Larger values of $COF_k(x)$ indicates that x is more anomalous. In the Figure (3.7) is presented the pseudo-code of Connectivity-based Outlier Factor Algorithm.

3.2.3 INFLuential measure of Outlierness by symmetric relationship (INFLO) Algorithm

When in the dataset there are more than one cluster and different clusters have different densities, the LOF fails to detect anomalies, so another approach has been proposed to solve this limitation of LOF. This method is called *INFLuential measure of Outlierness by symmetric relationship* and it uses the k nearest neighbors and reverse nearest neighbors of an point $x \in D$ to obtain a measure of outlierness. Given k, INFLO score is computed by means the following steps:

Algorithm COF

Require: k, D

Ensure: COF_k - COF score for each object in D

$COF_k = \emptyset$
for $p \in D$ **do**
    $N_k(p) = NULL$; $SBN(P) = \{P\}$; $SBTDist(p) = \emptyset$; $A_{N_k}(p) = 0$
    for $q \in D$
        if $|N_k(p)| < k$ **then**
            Add q in $N_k(p)$
        else
            Let $s^* \in N_k(p)$ be such that $dist(p, s^*) \geq dist(p, s)$ for all $s \in N_k(p)$;
            if $dist(p, s^*) > dist(p, q)$ **then**
                Replace $s^* \in N_k(p)$ by q
            end if
        end if
    end for
    $i = 1$; $NN(p) = N_k(p)$
    while $|NN(p)| > 0$ **do**
        $dist(e_i) = mini\{dist(s, t) | s \in SBN(p), t \in NN(p)\}$
        Move corresponding t from $NN(p)$ to $SBN(p)$
        $i++$
        Add $dist(e_i)$ to $SBTDist(p)$
    end while
    for $i = 1$ to k **do**
        $A_{N_k}(p) = A_{N_k}(p) + \frac{dist(e_i)*2(k+1-i)}{(k+1)*k}$
    end for
end for
for $p \in D$ **do**
    $COF_k(p) = \frac{|N_k(p)|*A_{N_k}(p)}{\sum_{o \in N_k(p)} A_{N_k}(o)}$
end for
return COF_k

Figure 3.7. **COF Algorithm**

- Compute $N_k(x)$ and $R_k(x)$ $\forall x \in D$.
- Compute the local density for each point as the inverse of the k-distance, i.e.:

$$den(x) = [d_k(x)]^{-1}.$$

- The k-influential space for x, denoted as $IS_k(x)$, is defined as $IS_k(x) = N_k(x) \cup R_k(x)$.
- The INFLO score is computed as:

$$INFLO_k(x) = \frac{\sum_{y \in IS_k(x)} den(y)}{|IS_k(x)|} \cdot [den(x)]^{-1}.$$

Algorithm INFLO
Require: k, D
Ensure: $INFLO_k$ - INFLO score for each object in D

$INFLO = \emptyset$
for $p \in D$ **do**
  $N_k(p) = \emptyset$; $RN_k(p) = \emptyset$;
  for $q \in D, q \neq p$ **do**
    if $|N_k(p)| < k$ **then**
      Add q in $N_k(p)$
    else
      Let $s^* \in N_k(p)$ be such that $dist(p, s^*) \geq dist(p, s)$ for all $s \in N_k(p)$;
      if $dist(p, s^*) > dist(p, q)$ **then**
        Replace $s^* \in N_k(p)$ by q
      end if
    end if
  end for
  $d_k(p) = max\{dist(p, s) | s \in N_k(p)\}$
  for $q \in N_k(p)$ **do**
    Add p in $RN_k(q)$
  end for
end for
for $p \in D$ **do**
  $IS_k(p) = N_k(p) \cup RN_k(p)$
  $INFLO_k(p) = \frac{d_k(p) * \sum_{o \in IS_k(p)} \frac{1}{d_k(o)}}{|IS_k(p)|}$
end for
return $INFLO_k$

Figure 3.8. **INFLO Algorithm**

Therefore, INFLO expands $N_k(x)$ to IS_k for each point $x \in D$ and compares x' density with the average density of points in IS_k, so INFLO improve its capacity to identify the anomalies by using the reverse neighborhood. In the Figure (3.8) is presented the pseudo-code of INFLuential measure of Outlierness by symmetric relationship Algorithm.

3.3 Rank Based Anomaly Detection Approaches

Rank Based Anomaly Detection is an another approach to identify anomalies based on mutual closeness of data point and its neighbors using rank instead of distance. Given k we consider $p \in D$ and $q \in N_k(p)$. The point q is close to p because it belong to k-neighborhood of p. If p and q are close to each other, then p is not outlier with respect to q and q is not outlier with respect to p. If q is a neighbor of p but p is not a neighbor of q, then p is an outlier with respect to q. If p is an outlier with respect to most

of its neighbors, then p should be declared to be an outlier. We present some of the most common rank-based algorithms used in anomaly detection: "*Rank Based Detection Algorithm*" and "*Rank with Averaged Distance Algorithm*".

3.3.1 Rank Based Detection Algorithm

When in the dataset there are clusters with different densities, *Rank Based Detection Algorithm* (RBDA) identifies, unlike the density based algorithms, the points at cluster boundary in a correct way. RBDA can be described by means of the following steps:

- For each point $x \in D$ is calculated $N_k(x)$.
- For each pair of points $(x, y) \in D$, it is calculated $r_x(y)$, i.e. the rank of y respect to x, as follow:
$$r_x(y) = |\{z : d(x, z) < d(x, y)\}|.$$
$r_x(y)$ measures how y is close to x; if there are fewer points between x and y, then y is close to x, i.e. $r_x(y)$ is very small.
- For each point $x \in D$ is calculated the RBDA score as follow:
$$RBDA_k(x) = \frac{\sum_{y \in N_k(x)} r_y(x)}{|N_k(x)|}.$$
Therefore, RBDA score that measures the outlierness of an object, is defined as the average rank with respect to its k-nearest neighbors.

In the Figure (3.9) is presented the pseudo-code of Rank Based Detection Algorithm.

3.3.2 Rank with Averaged Distance Algorithm

Rank with Averaged Distance Algorithm (RADA), for each point $x \in D$, adjusts the RBDA score by the averaged distance from its k nearest neighbors. Therefore, the RADA score is defined as:
$$RADA_k(x) = RBDA_k(x) \times \frac{\sum_{y \in N_k(x)} d(x, y)}{|N_k(x)|}.$$

Algorithm Rank-Based Detection Algorithm

Require: k, D

Ensure: $RBDA_k$ - RBDA score for each object in D

```
1: RBDA = ∅
2: for p ∈ D do
3:     N(p) = ∅; N_k(p) = ∅;
4:     for q ∈ D do
5:         if q ≠ p then
6:             Add q to N(p)
7:         end if
8:     end for
9:     Sort N(p) by dist(p, q) in ascending order
10:    d_k(p) = kth of N(p)
11:    tmp = 0; index = 0; rank = 0;
12:    for q ∈ N(p) do
13:        if dist(p, q) ≤ d_k(p) then
14:            Add q in N_k(p)
15:        end if
16:        index + +
17:        if dist(p, q) ≤ tmp then
18:            rank = index;
19:        end if
20:        r_p(q) = rank; tmp = dist(p, q);
21:    end for
22: end for
23: for p ∈ D do
24:    sumrank = 0;
25:    for q ∈ N_k(p) do
26:        sumranks+ = r_q(p)
27:    end for
28:    RBDA_k(p) = sumranks / |N_k(p)|
29: end for
30: return RBDA_k
```

Figure 3.9. **RBDA Algorithm**

Chapter 4

Ensemble Methods for Anomaly Detection

In the previous chapter we have presented different anomaly detection algorithms. A way to improve the performance of these algorithms, and successfully address the high false positive rates of individual algorithms is use the *ensemble methods* that combine the results of multiple algorithms often provide the best results. Ensemble methods can be categorized by component independence. The categorisation by component independence divides ensemble methods into two groups: the *independent ensemble* and *sequential ensemble*. In the independent ensemble the base learners are executed independently of each other, and their outputs are combined after all of the base learners have been applied in order to obtain a more robust model. In the sequential ensemble, the base learners are applied sequentially, meaning that the future algorithm will be dependent on previous algorithm.

4.1 Independent Ensemble Methods for Anomaly Detection

In the ensemble methods, the final decision is jointly decided by multiple base components. In the independent ensemble methods, each algorithm returns an anomaly score for each point in D. The observations that receive higher score are considered more anomalous. The range and distributions of scores may be substantially different for different algorithms, thus is necessary, before combining scores, the normalization of individual scores. Normalization assigns a normalized score of 0 to the least anomalous observation and a normalized score of 1 to the most anomalous observation. Let $\alpha_i(p)$ be the normalization

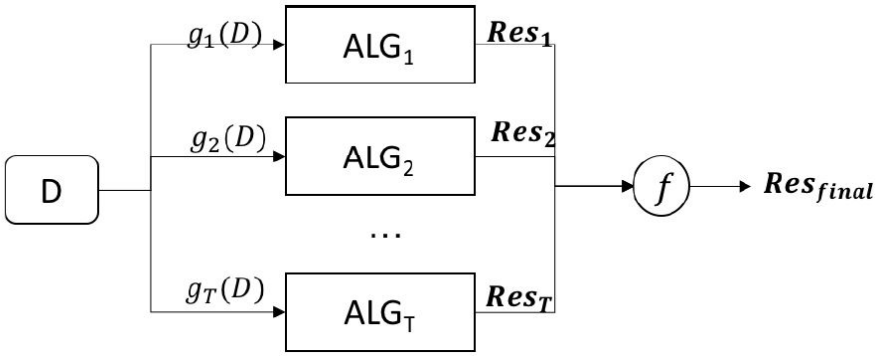

Figure 4.1. Independent Ensemble Method for Anomaly Detection.

anomaly score of $p \in D$ with respect to algorithm i. There are various combinations that can be used to combine all the outputs obtained from components. The Figure(4.1) presents an scheme of the independent ensemble approach. In particular there are T base components, each of which could be a different algorithm, or the same algorithm with different parameter settings. Let g_i be a data transform function applied to a given dataset D. It return an output that will be the input of the i-th algorithm. Each base component outputs a vector $\boldsymbol{Res_i}$ indicates the results made from component i. In this vector there are the anomaly scores of the points. A combination function f is applied to $\boldsymbol{Res_i}, i = 1, ..., T$ to make the final decision $\boldsymbol{Res_{final}}$.

A important step in building ensemble methods is to combine the base learners. There are many combination approaches that can be used to combine the results of multiple algorithms and we present some of the most common. These combination methods differ in the way to seek a consensus among algorithms. Furthermore they can be categorized, with respect to the type of combination used to obtain the final result, into two groups: *Score Based Combination Methods* and *Rank Based Combination Methods*.

Score Based Combination Methods

- ***The score averaging method***: combines normalized anomaly scores for each object. Let $\alpha_i(x)$ be the normalization anomaly score of the point $x \in D$ with respect to algorithm i, then the averaged normalized score of x over all T individual algorithms is given by:

$$\alpha(x) = \frac{1}{T} \sum_{i=1}^{T} \alpha_i(x).$$

  The point with the highest value of α is most anomalous and it is ranked as first.

- ***The maximum score combination method***: selects, for each point, the maximum score from the T individual algorithms and it is defined as follow:

$$\alpha(x) = \underset{i=1,..,T}{max}\alpha_i(x).$$

Rank Based Combination Methods

- ***The minimum ranking method***: selects, for each point, the minimum rank as final output, instead of using the score output. The anomalous rank of a point $x \in D$, assigned by algorithm i, is given by:

$$r_i(x) = |D| - \{y \quad | \quad \alpha_i(y) < \alpha_i(y)\},$$

where $|D|$ is the number of points in D. More small is the rank and more anomalous is the point. The minimum rank method is defined as follow:

$$rank(x) = \underset{i \leq i \leq T}{min} r_i(x).$$

Therefore, if the point x is considered the most anomalous by at least one algorithm, then this method consider it the most anomalous point.

- ***The averaging ranking method***: considers the mean value of ranking over all T individual algorithms and it is given by:

$$rank^{'}(x) = \frac{1}{T}\sum_{i=1}^{T} r_i(x).$$

More is small the $rank^{'}(x)$, more the point x is considered anomalous.

Majority Voting Rule Methods

Another approach is *majority voting* and it is one of the most popular and intuitive combination methods. In the dataset the number of outliers should be very small, so we consider the ranked top $\tau\%$ as the true outliers for each individual algorithm. Therefore the decision of each object x from the i-th component is:

$$H_i(x) = \begin{cases} 1, & if \quad r_i(x) \leq \tau\% \times |D| \\ 0, & otherwise. \end{cases}$$

The final vote for each point x is given by:

$$V(x) = \sum_{i=1}^{T} H_i(x).$$

The final decision to establish if x is an outlier is:

$$H(x) = \begin{cases} 1, & V(x) > \frac{T}{2} \\ 0, & otherwise. \end{cases}$$

However, the ensemble's individual base algorithms often do not have equal performance. Hence, considering them with equal weights might be inappropriate. A more suitable solution is to use different weights for the performances of the individual algorithms using the weighted majority voting technique.

Weighted Majority Voting Methods

The *weighted majority voting* assumes that some base learners in the ensemble method are more skilful than others, and their results are given more priority when computing the final ensemble prediction. The majority voting assumes that all the base learners are equally important and their results are treated equally when calculating the final ensemble result. Instead, the weighted majority voting assigns more weight to the algorithms that agree more with the majority. The based idea is that if a base learner find the same set of outliers as the majority, it have a higher weight than the one that often disagrees with the majority. We consider a weight w_i for each base algorithm i given by:

$$w_i = \frac{\sum_{x \in D} I(H_i(x), H(x))}{\sum_{x \in D} H(x)},$$

where $H_i(x)$ is the decision of i-th algorithm, while $H(x)$ is the decision reached by the majority of the base learners and they are defined as in the majority voting. Instead, the function I is defined as follow:

$$I(x, y) = \begin{cases} 1, & if \quad x = y \\ 0, & otherwise. \end{cases}$$

Thus, the base learners that agree more often with the majority will have higher weight than the others.

4.2 Sequential Ensemble Methods for Anomaly Detection

Sequential Ensemble Methods for anomaly detection is an other approach to building an ensemble method. In the independent ensemble approach an important assumption is

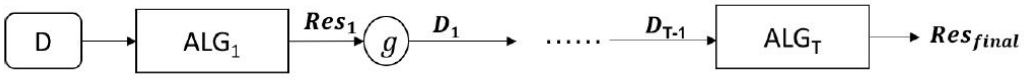

Figure 4.2. **Single-layer sequential ensemble methods**

that the base learners have to be independent of each other and it might be unrealistic. Introducing randomness into ensemble methods decreases the dependence between components, however there is inherent dependence between the base algorithms because they are perturbation of the same general idea. Sequential approach overcomes this problem because considers the dependence between the components but is very importance to use different algorithms that work in sequence. The diversity of the base learners is fundamental because the next component tries to resolve the error produced by the previous algorithm. But, if they are similar then they will make a similar mistake. Thus, the diversity improve the performances of the ensemble methods. The sequential ensemble methods can be categorized into two classes: *single-layer sequential* and *multi-layer sequential*.

Single-layer Sequential Ensemble Methods

In the single-layer sequential ensemble methods the first algorithm is applied to the entire dataset and the output of this is the input of the next algorithm and it proceeds in this way for all the algorithms in sequence. In the Figure (4.2) is shown the single-layer sequential approach using multiple base algorithms. Let T be the number of base algorithms that are used. At each step, the algorithm ALG_i is applied and it generates an intermediate result Res_i. To generate the input D_i for the next algorithm ALG_{i+1} a transformation function can be applied to Res_i.

Multi-layer Sequential Ensemble Methods

In the single-layer sequential approach the first algorithm is applied to the entire dataset D, and then the results are refined in later steps sequentially. Instead, in the multi-layer methods, intermediate results are generated from a previous ensemble model, and then another algorithm combines these results to obtain the final decision. In the Figure (4.3) is shown the multi-layer sequential approach, in particular there are two-layer structure. The first layer is equal to independent ensemble approach described in the previous section. There are T independent base algorithms, denoted with ALG_{1i} for $i = 1, ..., T$, that are applied on the same dataset D. Let f be a consensus function, it can be applied to generate the first-layer decision combining the results from multiple base algorithms, thus the first-layer can generate diverse ensembles. Then, a data transforma-

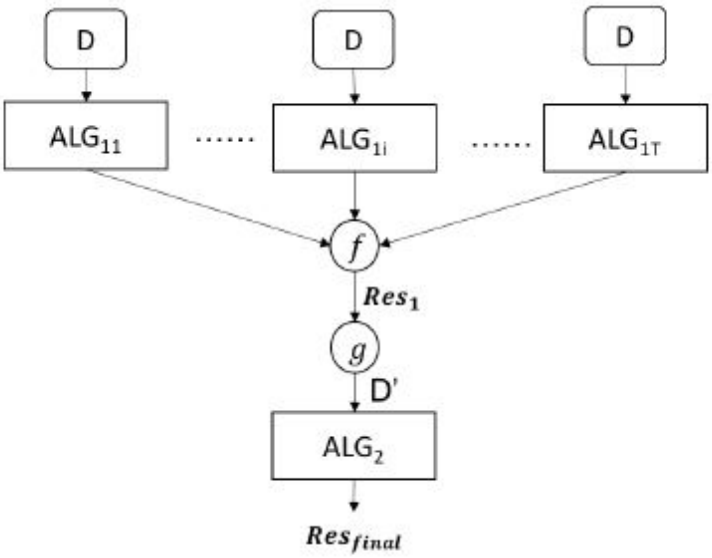

Figure 4.3. **Two-layer sequential ensemble methods**

tion function g is applied to the result obtained from f and the output generated from g can be, for example, a new data sample $D^{'}$. In the second layer the algorithm ALG_2 is applied to $D^{'}$ and it generates the final decision Res_{final}.

4.2.1 Sequential ensemble method with two algorithms

In this sequential ensemble method two algorithms are combined in sequence. The idea is that similar algorithms tend to make similar errors, so applying in sequence two substantially different algorithms, the second algorithm can "correct" the errors of the previous one and thus provide better results.

The method consists of executing the first algorithm in the original dataset D to extract a subset of D that will be used in the second algorithm . In particular, in the first phase we consider an anomaly detection algorithm that is applied to the entire dataset D which returns the ranks of all observations, calculated using anomaly scores. The subset D_β is constructed by considering the β fraction of D that the first algorithm considers more abnormal. Finally, the second algorithm calculates the anomaly scores of all objects in D with reference to D_β. This method is described in the Figure (4.4).

Algorithm Sequential-1 Algorithm
Data: Dataset D

Result: a vector of scores $\mathbf{H}$ associated with each of the objects in D

Score vector $\mathbf{H_A}$ is obtained by applying algorithm A on D;

Rank vector $\mathbf{R_A} = \{r_A | \forall x \in D\}$; objects are sorted in decreasing order of H_A.

$D_\beta = \{x | r_A(x) < \beta \cdot |D|\}$; i.e. retrieve the top ranked data.

for all $x \in D$ **do**
    $H(x)$ =calculate the anomaly score of x by applying algorithm B on dataset $\{x\} \cup D_\beta$
end

return H

Figure 4.4. **Sequential-1 Algorithms: sequential ensemble method with two algorithms**

4.2.2 Sub-sampling and Sequential Method

An approach to introduce randomness into ensemble methods is the *sub-sampling*. Furthermore, it was demonstrated that the anomaly detection methods perform better when applied to a sub-sample of the entire dataset. The sub-sampling approaches select a random sample from the dataset D and compute the outlier score for each point in D with respect to the data in the selected sample. This process is repeated multiple times in a way that for each point is computed the average score outlier. For example, an sub-sampling approach can be consider the D_β dataset that is the dataset obtained by retaining the most anomalous fraction β of D with respect to an anomaly algorithm, i.e., those that suspected to contain most or all anomalies. Then is considered a percentage γ from D_β and it is compute the outlier score for each point $x \in D$ using another algorithm with respect to the sub-sample. This process is repeated, for example, T times, and then is calculated the average scores that are used in the final anomaly step. In the Figure (4.5) is described the pseudo-code of this method.

We select the top ranked $\beta \cdot |D|$ points as those suspected to contain anomalies. Thus, β is a essential parameter and it is important to understand how to choose a suitable value for β. Let ALG_i be the i-th anomaly detection algorithm and let $\alpha_i(x)$ the real valued score for each point $x \in D$ generated from ALG_i. Therefore, the rank of the point x is given by:

$$r_i(x) = |D| - |\{y \quad | \quad \alpha_i(y) < \alpha_i(x)\}|.$$

Algorithm Sequential-2 Algorithm
Data: Dataset D

Result: a vector of scores $\mathbf{H}$ associated with each of the objects in D

Score vector $\mathbf{H_A}$ is obtained by applying algorithm A on D;

Rank vector $\mathbf{R_A} = \{r_A | \forall x \in D\}$; objects are sorted in decreasing order of H_A.

$D_\beta = \{x | r_A(x) < \beta \cdot |D|\}$; i.e. retrieve the top ranked data.

for all $x \in D$ **do**
    for $i = \{1, ..., T\}$ **do**
        D_γ =randomly pick $\gamma \cdot |D_\beta|$ objects from D_β;
        $H_i(x)$ =apply algorithm B on dataset $\{x\} \cup D_\gamma$; i.e. obtain the score of x from the i^{th} iteration;
    end
end

for $\forall x \in D$ **do**
    $H(x) = \frac{1}{T} \sum_{i=1}^{T} H_i(x)$;
end

return H

Figure 4.5. **Sub-sampling and sequential algorithm**

Indicating with β the fraction that we want to consider, we expect that an accurate algorithm ALG_i satisfy the following inequality:

$$Pr(r_i(x) > |D| \cdot \beta \quad | \quad x \in \mathcal{O}) < f(\beta),$$

where $\mathcal{O}$ is the set of outliers, $|D|$ is the number of point in the dataset D and f is a function of β. The number of outliers should be decreasing as β increasing because most outliers should be ranked in the topped percentage.

Chapter 5

Experiments

The goal of our work is to show how the implemented ensemble methods, both independent and sequential, return better performances in anomaly detection than the basic components with which they are built. In particular, for our experiments we used three datasets **KDD CUP 99**, **Simargl2022** and **IoT Network Intrusion Dataset**.

- **KDD CUP 99**: is a benchmark dataset widely used in intrusion detection. It consists of 42 continuous and discrete attributes. For our experiments we used 38 features. The remaining 4 features were deleted from the dataset. In the appendix of this paper are present Tables A.1, A.2 and A.3. Tables A.1 and A.2 contain the name, description and type of the features effectively used in the experiments, while Table A.3 contains the deleted features. The observations of the dataset are labeled with the specific attribute "*label*", which assumes value "*Normal*" if the observation is an inlier, otherwise, if the observation is an outlier, it has the value corresponding to the attack on the network. Network attacks can be of various types.

  For the continuous features of the dataset, the mean and standard deviation were calculated to understand their distribution. The obtained values are shown in the Table 5.1.

- **Simargl2022**: is a dataset where observations come from a real-world academic network. Real traffic was collected and, after performing a series of attacks, the dataset was assembled. It consists of 44 network features. Also in this case, the features deleted were 4. As in the previous dataset, the appendix shows Tables A.4 and A.5 containing the features used in the experiments and the eliminated features, respectively. In this case the anomalies in the dataset all belong to a single attack. Dataset observations are labeled with the "*attack*" feature, which assumes value "*None*" for inliers, while "*PortScan*" for outliers.

	Mean	Std
duration	47.979302	707.746472
src_bytes	302.610296	988218.101050
dst_bytes	868.532425	33040.001252
hot	0.034519	0.782103
num_compromised	0.010212	1.798326
num_root	0.011352	2.012718
num_file_creations	0.001083	0.096416
num_access_files	0.001008	0.036482
Count	332.285690	213.147412
srv_count	292.906557	246.322917
serror_rate	17.668666	38.071696
srv_serror_rate	17.660881	38.101658
rerror_rate	5.743341	23.162347
srv_rerror_rate	5.771894	23.214698
diff_srv_rate	79.154734	38.818949
srv_doff_host_rate	2.098239	8.220549
dst_host_count	2.899680	14.239747
dst_host_srv_count	232.470778	64.745380
dst_host_same_srv_rate	188.665670	106.040447
dsr_host_same_srv_rate	75.377970	41.078098
dst_host_diff_srv_rate	3.090573	10.925911
dst_host_same_src_port_rate	60.193476	48.130925
dst_host_srv_diff_host_rate	0.668350	4.213287
dst_host_serror_rate	17.675396	38.059310
dst_host_srv_serror_rate	17.644262	38.091945
dst_host_rerror_rate	5.811761	23.058951
dst_host_srv_rerror_rate	5.741167	23.014032

Table 5.1. **KDD CUP 99:** mean and variance for each feature.

As with the previous dataset, we calculated the mean and standard deviation for all continuous features. The obtained values are shown in the Table 5.2.

- **IoT Network Intrusion Dataset**: is a dataset containing data about network attacks. The dataset consists of 86 features, but the features used in the experiments are 68. As in previous cases, the appendix shows Tables A.6, A.7 and A.8. The first two tables contain the features used in the experiments, while the third table contains the eliminated features. Dataset observations are labeled with the "*Label*" feature, which assumes value "*Normal*" for inliers, while "*Anomaly*" for outliers.

  Also in this case we calculated the mean and standard deviation for all continuous features. The obtained results are shown in the Table 5.3.

We have analyzed histograms of features of dataset, however only some of them, useful for discussing the obtained results in experiments, are reported in the following sections. For each dataset we considered 1150 observations, divided as follows: 1000 inliers and 150

	Mean	Std
L4_SRC_PORT	4.179941e+04	1.996662e+04
L4_DST_PORT	4.739815e+03	1.391079e+04
FIRST_SWITCHED	1.647522e+09	1.469765e+04
FLOW_DURATION_MILLISECONDS	6.444856e+03	2.221613e+04
LAST_SWITCHED	1.647522e+09	1.469747e+04
PROTOCOL	1.041372e+01	5.985748e+00
TCP_FLAGS	1.105820e+01	1.718039e+01
TCP_WIN_MAX_IN	1.323570e+04	2.534137e+04
TCP_WIN_MAX_OUT	1.155426e+04	2.400037e+04
TCP_WIN_MIN_IN	1.319567e+04	2.533628e+04
TCP_WIN_MIN_OUT	1.151889e+04	2.399165e+04
TCP_WIN_MSS_IN	3.312921e+02	6.113026e+02
TCP_WIN_SCALE_IN	1.453568e+00	3.010768e+00
TCP_WIN_SCALE_OUT	1.388693e+00	3.081813e+00
SRC_TOS	7.624096e+00	3.198905e+01
DST_TOS	8.930120e+00	3.495344e+01
TOTAL_FLOWS_EXP	3.575022e+08	9.717543e+05
MIN_IP_PKT_LEN	0.000000e+00	0.000000e+00
MAX_IP_PKT_LEN	0.000000e+00	0.000000e+00
TOTAL_PKTS_EXP	0.000000e+00	0.000000e+00
TOTAL_BYTES_EXP	0.000000e+00	0.000000e+00
IN_BYTES	2.417737e+03	2.559013e+04
IN_PKTS	1.632104e+01	2.815912e+02
OUT_BYTES	5.019650e+04	1.284571e+06
OUT_PKTS	4.144903e+01	9.047939e+02

Table 5.2. **Simargl2022:** mean and variance for each feature.

outliers. Therefore, contamination of anomalies is 13,04%.

The base learners used in the experiments and described in the chapter 3 are 6, specifically :

- two density-based learners (LOF, COF)
- two distance-based learners (KNN, LOCI)
- two rank-based learners (RBDA, RADA).

For each algorithm, the number of k neighbors (which can assume one of the following values: $5, 10, 15, 20, 25$) has been chosen to obtain the best performance. The datasets were divided as follows: 70% training set, 30% testing set.

5.1 Metrics for Measurement

To evaluate the performance of the algorithms, the most common metrics used are *precision*, *recall* and *Rank-Power*.

	Mean	Std		Mean	Std
Src_Port	35012.5309	24721.3771	**Bwd_IAT_Max**	486.9912	2752.7515
Dst_Port	16391.4648	17554.5528	**Bwd_IAT_Min**	429.1097	2178.4459
Flow_Duration	635.7967	3497.7353	**Fwd_Header_Len**	22.4988	41.5412
Tot_Fwd_Pkts	1.6759	4.3112	**Bwd_Header_Len**	33.7346	40.2906
Tot_Bwd_Pkts	1.4688	1.2197	**Fwd_Pkts/s**	50336.8692	164455.6835
TotLen_Fwd_Pkts	571.0499	1162.1394	**Bwd_Pkts/s**	27834.4737	78501.6543
TotLen_Bwd_Pkts	929.8016	1732.1344	**Pkt_Len_Min**	511.9242	654.0406
Fwd_Pkt_Len_Max	392.6978	619.6956	**Pkt_Len_Max**	700.4750	696.8593
Fwd_Pkt_Len_Min	348.3090	588.2846	**Pkt_Len_Mean**	634.0147	652.3351
Fwd_Pkt_Len_Mean	373.7538	596.6703	**Pkt_Len_Std**	102.6116	243.2846
Fwd_Pkt_Len_Std	28.1773	227.8352	**Pkt_Len_Var**	69716.4849	243.2846
Bwd_Pkt_Len_Max	681.7994	695.0233	**Down/Up_Ratio**	0.3642	0.4990
Bwd_Pkt_Len_Min	588.9856	683.0051	**Pkt_Size_Avg**	915.7478	948.7259
Bwd_Pkt_Len_Mean	637.4560	669.5989	**Fwd_Seg_Size_Avg**	373.7538	596.6703
Bwd_Pkt_Len_Std	63.0282	227.8352	**Bwd_Seg_Size_Avg**	637.4560	669.5989
Flow_Pkts/s	78171.3429	220072.3732	**Subflow_Fwd_Pkts**	1.6759	4.3112
Flow_IAT_Mean	483.7470	1893.3819	**Subflow_Fwd_Byts**	571.0499	1162.1394
Flow_IAT_Std	63.1740	1160.0533	**Subflow_Bwd_Pkts**	1.4688	1.2197
Flow_IAT_Max	566.1531	2867.1355	**Subflow_Bwd_Byts**	929.8016	1732.1344
Flow_IAT_Min	443.3637	1719.8049	**Init_Bwd_Win_Byts**	5884.3802	11532.1337
Flow_IAT_Tot	102.2180	2216.9550	**Fwd_Act_Data_Pkts**	1.5102	4.3339
Fwd_IAT_Mean	52.5464	1212.3582	**Active_Mean**	3.7666	68.0844
Fwd_IAT_Std	27.7055	959.3051	**Active_Std**	0.3535	20.7294
Fwd_IAT_Max	85.0741	1936.1797	**Active_Max**	4.2512	88.9602
Fwd_IAT_Min	34.6997	1000.6796	**Active_Min**	3.4641	64.1298
Bwd_IAT_Tot	517.0968	3148.1876	**Idle_Mean**	502.7995	2113.5437
Bwd_IAT_Mean	447.2275	2243.0970	**Idle_Std**	52.4348	1153.5234
Bwd_IAT_Std	28.8076	808.9099			

Table 5.3. **IoT Network Intrusion Dataset:** mean and variance for each feature.

Let D be a dataset with n observations and it contains d_t true outliers. Given a anomaly detection algorithm, we suppose that it identifies $m > 0$ outliers in D but the number of true anomalies among m instances is $m_t (\leq m)$. The *precision* is defined as follow:

$$\mathbf{Pr} = \frac{m_t}{m}.$$

This metric measures the number of true outliers with respect to the number of outliers identified by the algorithm. The *recall* is defined as follow:

$$\mathbf{Re} = \frac{|m_t|}{|d_t|}.$$

The recall measures the accuracy of the algorithm and if it is equal 1.0 then all true outliers are discovered by algorithm.

Precision and recall are not sufficient to capture completely the efficiency of an algorithm, in particular when comparing algorithms that find different numbers of outliers. One algorithm may indicate an anomaly as the most suspicious while another algorithm may indicate it as the lest suspicious, but the values of precision and recall remain the same. Therefore, an algorithm will be considered more efficient if the true outliers occupy

top position and the inliers are among the least suspicious instances. This idea is captured by *Rank-Power* metric that is defined as:

$$\mathbf{RP} = \frac{m_t(m_t+1)}{2\sum_{i=1}^{m_t} R_i},$$

where R_i is the rank of i-th true outlier in the sorted list of most suspicious objects. The value of Rank-Power is 1 when all n true outliers are in top n position. For a fixed value of m, larger values of all three metrics imply better performance.

Another metric that we mainly use is *Area Under Curve (AUC)*. The AUC is the area under the ROC curve and it is used to reduce the ROC curve to a single number. The *Receiver Operating Characteristic* (ROC) curve is the plot of true positive rate (TPR) against false positive rate (FPR). Let $\mathcal{O}$ and $\mathcal{I}$ be the anomaly class (outliers) and the normal class (inliers), respectively. Let $\hat{\mathcal{O}}$ and $\hat{\mathcal{I}}$ be the predicted outliers and predicted inliers, respectively. Then, the true positive rate and the false positive rate are defined as:

$$TPR = \frac{|\hat{\mathcal{O}} \bigcap \mathcal{O}|}{|\mathcal{O}|} \qquad and \qquad FPR = \frac{|\hat{\mathcal{O}} \bigcap \mathcal{I}|}{|\mathcal{I}|}$$

The AUC, i.e. the area under such curve, will also always be between 0 and 1, where 1 is the best evaluation that an algorithm can achieve, meaning it sorted all the outliers prior to all the inliers.

To compare the various methods implemented, we have used precision and recall for the positive class, corresponding to the inliers, and for the negative class, corresponding to the outliers. Furthermore, we reported the false positive rate and confusion matrix for each method.

5.2 Results anomaly detection algorithms: KDD CUP 99 dataset

For the first dataset, the obtained values with the base learners are shown in the Tables 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9. We set $k = 25$, $k = 20$, $k = 25$, $k = 25$, $k = 10$ and $k = 5$ for the LOF, COF, KNN, RBDA, RADA and LOCI, respectively. We observe that, for KDD CUP 99 dataset, the base algorithm RBDA achieves the best performance among all the individual base learners.

LOF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	283	28	0.9099	0.9218	0.2941	0.2631	0.7368
NC	24	10					

Table 5.4. KDD CUP 99 dataset - LOF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

COF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	273	25	0.9161	0.8892	0.2741	0.3421	0.6578
NC	34	13					

Table 5.5. KDD CUP 99 dataset - COF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 20$.

KNN						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	287	28	0.9111	0.9348	0.3333	0.2631	0.7368
NC	20	10					

Table 5.6. KDD CUP 99 dataset - KNN algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

RBDA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	291	9	0.97	0.9478	0.6444	0.7631	0.2368
NC	16	29					

Table 5.7. KDD CUP 99 dataset - RBDA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

5.3 Results independent ensemble methods: KDD CUP 99 dataset

To build the independent ensemble methods we used three different combinations: the maximum score method, the score averaging method and the majority vote. We have considered all the possible combinations that can be obtained with the six base learners

RADA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	272	28	0.9066	0.8859	0.2222	0.2631	0.7368
NC	35	10					

Table 5.8. KDD CUP 99 dataset - RADA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 10$.

LOCI						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	300	30	0.9090	0.9771	0.5333	0.2105	0.7894
NC	7	8					

Table 5.9. KDD CUP 99 dataset - LOCI algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 5$.

used. However, we only report and explore some examples.

We now analyze the results obtained with the independent ensemble methods for the first dataset. In the first example (5.10) we can see that the performance of the independent ensemble method is better than that obtained with the base learners with which it is built. In particular we can notice that by combining, with the score averaging method, LOF algorithm and RADA algorithm we have a precision, both of positive and of negative class, greater than that obtained with the single algorithms, by going from 0.9099 to 0.9381 and from 0.2941 a 0.5, respectively, compared with LOF algorithm and by going from 0.9066 to 0.9381 and from 0.2222 a 0.5, respectively, compared with RADA algorithm. The recall, for both classes, also improves, going from 0.9218 to 0.9381 for positive class and from 0.2631 to 0.5 for negative class, compared with LOF algorithm and going from 0.8859 to 0.9381 for positive class and from 0.2631 to 0.5 for negative class, compared with RADA algorithm. The false positive rate is equal to 0.5 and therefore decreases both with respect to the LOF algorithm and with respect to the RADA algorithm, whose value is, for both, 0.7368. Therefore, the independent ensemble method constructed by combining LOF and RADA algorithms, with the score averaging method, allows to better classify inliers and outliers than individual anomaly detection algorithms.

For a more in-depth analysis we analyzed the features both for the observations classified, by the LOF-RADA independent ensemble method, as inliers and for those classified as outliers. Then, we computed the mean and standard deviation for both inliers and outliers. The values obtained, for each feature and for each class, are shown in the Table

ENSEMBLE: LOF-RADA - COMBINATION: AVERAGE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	288	19	0.9381	0.9381	0.5	0.5	0.5
NC	19	19					

Table 5.10. KDD CUP 99 dataset - independent ensemble method by combining LOF and RADA algorithms with the score averaging: precision and recall for positive and negative class, false positive rate and confusion matrix.

	Mean_Inliers	**Std_Inliers**	**Mean_Outliers**	**Std_Outliers**
duration	75.983713	573.539084	188.052632	1153.908656
src_bytes	477.576547	771.139198	1769.973684	8801.678594
dst_bytes	2008.302932	4331.415558	11297.921053	45310.014972
hot	0.000000	0.000000	0.052632	0.324443
num_compromised	0.000000	0.000000	0.026316	0.162221
num_root	0.003257	0.057073	0.000000	0.000000
num_file_creations	0.000000	0.000000	0.026316	0.162221
num_access_files	0.000000	0.000000	0.052632	0.226294
Count	39.394137	122.888617	166.236842	209.258205
srv_count	42.280130	122.645857	129.026316	215.699677
serror_rate	0.107492	1.148248	7.894737	27.327631
srv_serror_rate	0.055375	0.450340	7.894737	27.327631
rerror_rate	4.967427	21.606226	15.789474	36.953702
srv_rerror_rate	4.902280	21.590898	15.789474	36.953702
diff_srv_rate	99.934853	1.141460	72.342105	38.642699
srv_doff_host_rate	0.130293	2.282921	15.052632	31.037856
dst_host_count	12.146580	26.725083	8.421053	27.364044
dst_host_srv_count	154.192182	102.147971	227.315789	61.962982
dst_host_same_srv_rate	215.885993	72.527187	115.605263	114.199574
dsr_host_same_srv_rate	88.459283	25.575995	50.815789	46.121376
dst_host_diff_srv_rate	2.804560	12.611370	10.131579	23.715853
dst_host_same_src_port_rate	15.162866	30.187641	42.121579	47.643479
dst_host_srv_diff_host_rate	2.042345	3.467142	0.263158	1.155111
dst_host_serror_rate	0.104235	0.648438	7.921053	27.320304
dst_host_srv_serror_rate	0.074919	0.349027	7.921053	27.320304
dst_host_rerror_rate	4.840391	20.707201	15.894737	36.913185
dst_host_srv_rerror_rate	4.814332	20.566679	15.894737	36.913185

Table 5.11. **KDD CUP 99 - Independent LOF-RADA:** mean and variance for each feature for each class (inliers and outliers).

5.11. We can see that for some features, such as "*diff_srv_rate*", the values obtained are different, while for other features, such as "*dst_host_count*", the values are very similar and therefore it is more complicated to distinguish them. Furthermore, we noticed that, in the first dataset, the features with the most interesting behavior are "*hot*" and "*num_file_creations*" whose histograms are reported in the Figure 5.1.

In both histograms we note that the value 0 is the most frequent but, there is also another value that stands out from the rest. For this reason we wondered how the independent ensemble method LOF-RADA, built by combining the results with the score

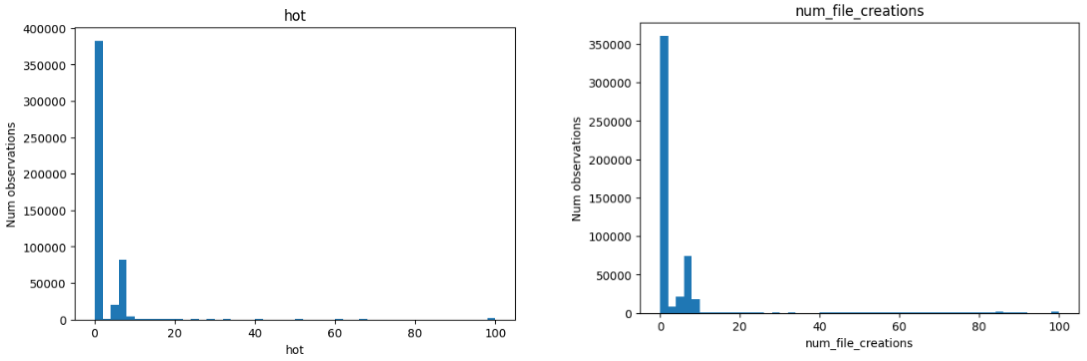

Figure 5.1. **KDD CUP 99 dataset:** histogram *hot* feature and histogram *num_file_creations* feature.

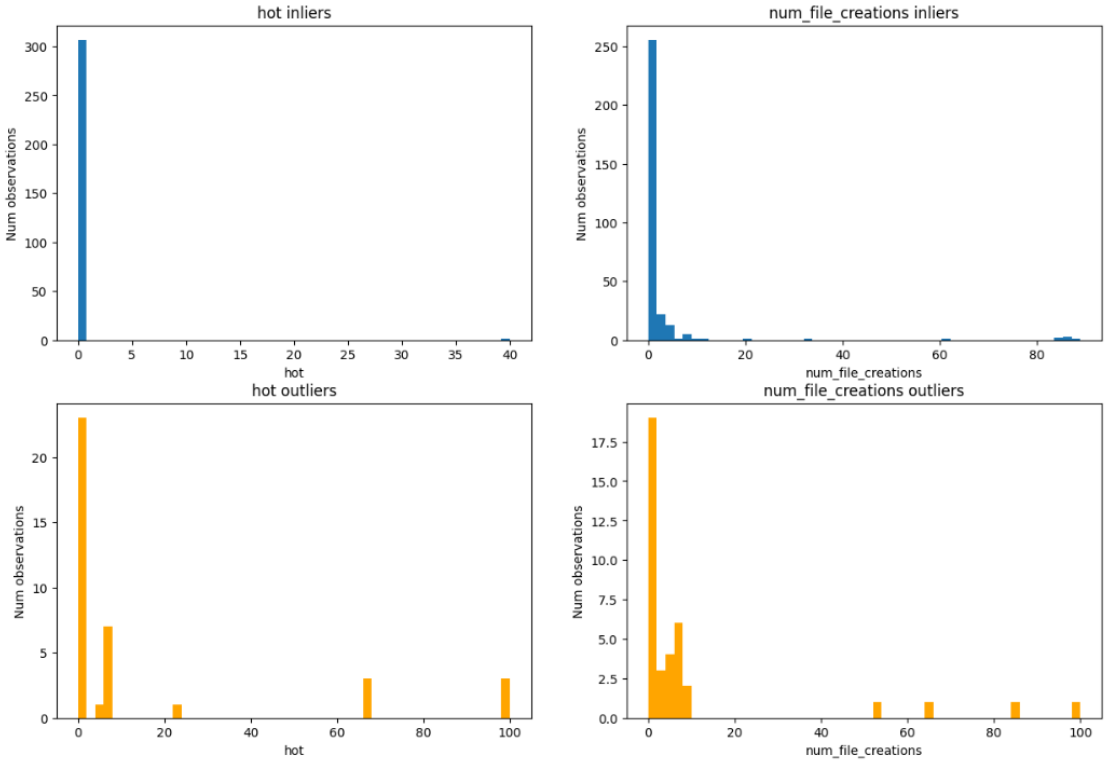

Figure 5.2. **KDD CUP 99 dataset - Independent LOF-RADA:** histograms of *hot* and *num_file_creations* features (inliers in blue, outliers in orange).

averaging, identified the observations of the two peaks, and in particular of the second one. Therefore, we plotted the histograms of the two features for the anomalies and for the inliers identified by this method and we noticed that the value 0, i.e. the most frequent, is

	Mean	Std
duration	0.0	0.0
src_bytes	1032.0	0.0
dst_bytes	0.0	0.0
hot	0.0	0.0
num_compromised	0.0	0.0
num_root	0.0	0.0
num_file_creations	0.0	0.0
num_access_files	0.0	0.0
Count	511.0	0.0
srv_count	511.0	0.0
serror_rate	0.0	0.0
srv_serror_rate	0.0	0.0
rerror_rate	0.0	0.0
srv_rerror_rate	0.0	0.0
diff_srv_rate	100.0	0.0
srv_doff_host_rate	0.0	0.0
dst_host_count	0.0	0.0
dst_host_srv_count	255.0	0.0
dst_host_same_srv_rate	255.0	0.0
dsr_host_same_srv_rate	100.0	0.0
dst_host_diff_srv_rate	0.0	0.0
dst_host_same_src_port_rate	100.0	0.0
dst_host_srv_diff_host_rate	0.0	0.0
dst_host_serror_rate	0.0	0.0
dst_host_srv_serror_rate	0.0	0.0
dst_host_rerror_rate	0.0	0.0
dst_host_srv_rerror_rate	0.0	0.0

Table 5.12. **KDD CUP 99 dataset - Independent LOF-RADA:** mean and standard deviation of false positives.

present in both classes for both the "*hot*" feature and the "*num_file_creations*" and that this independent ensemble method identifies as outliers the observations having as value of features the one corresponding to the second peak.

This can be seen in the histograms of the two features in Figure 5.2. The histograms in blue represent the distributions of inliers, while those in orange represent the distributions of outliers. To understand the limits of this method we tried to understand why it failed to identify some anomalies. First we noticed that all the anomalies not identified by the method have the same values of features. Indeed, in the Table 5.12, containing the mean and the standard deviation of the false positives, we can see that the standard deviation for all the features is 0. Analyzing the features individually, we noticed that the 19 observations, identified as inliers by the method but which are actually anomalies, have "*Count*=511" and "*srv_count*=511" and they are the only observations that assume this value for these features, as shown in the histograms in the Figure 5.3.

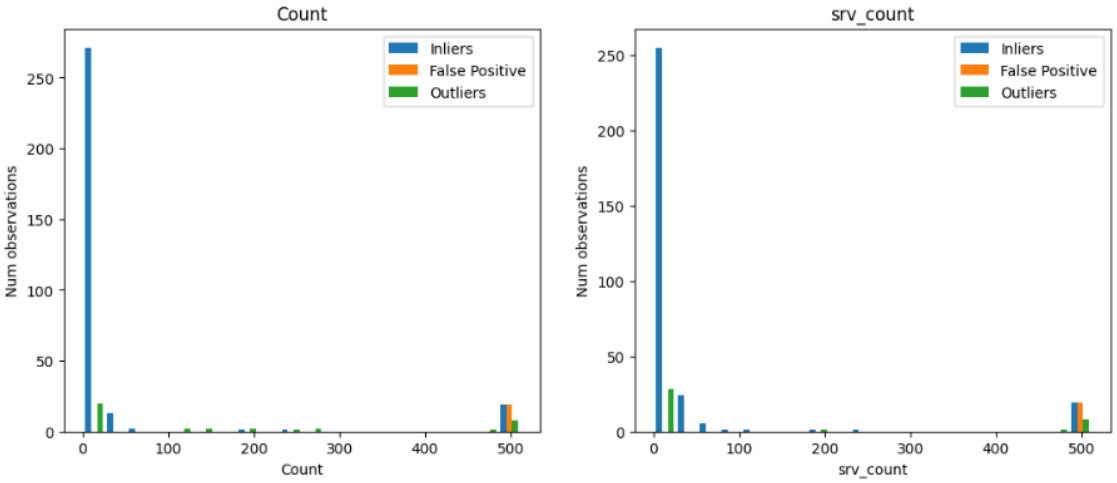

Figure 5.3. **KDD CUP 99 dataset - Independent LOF-RADA:** histograms of *Count* and *srv_count* features (inliers in blue, false positives in orange and outliers in green).

For all the other features we have been seen that the assumed value are the most frequent and they are present in both classes. The only feature that could highlight that these observations are outliers was "*src_bytes*". In fact, the inliers identified by the method assume the value 1032 only in correspondence with these false positives, while among the anomalies it is a value present several times. This situation makes us understand that the method has difficulty in identifying the anomalies that assume the most frequent value in most of the features. Furthermore, since these 19 observations are all the same, the method probably fails to identify them correctly because it considers these objects quite frequent.

In the second example (5.13) we have combined COF and RBDA with the maximum score method. We obtain a precision, both of positive and of negative class, greater than that obtained with the COF. Furthermore, compared with the COF algorithm, we note that the precision improves going from 0.9161 to 0.9675 and going from 0.2741 to 0.7567 for positive and negative class, respectively. The recall, for both classes, improves, going from 0.8892 to 0.9706 for positive class and from 0.3421 to 0.7368 for negative class, while the false positive rate decreases significantly by going from 0.6578 to 0.2631. Furthermore, for the first dataset, we have better performance with independent ensemble method built by combining, with the maximum score, COF and RBDA than with the COF algorithm.

Instead, we note that compared to the RBDA algorithm the precision of positive class and the recall of negative class decrease slightly by going from 0.97 to 0.9675 and from 0.7631 to 0.7368 respectively, while the precision of negative class and the recall of positive

ENSEMBLE: COF-RBDA - COMBINATION: MAXIMIZATION						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	298	10	0.9675	0.9706	0.7567	0.7368	0.2631
NC	9	28					

Table 5.13. KDD CUP 99 dataset - independent ensemble method by combining COF and RBDA algorithms with the maximum score: precision and recall for positive and negative class, false positive rate and confusion matrix.

ENSEMBLE: LOF-RBDA - COMBINATION: AVERAGE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	297	10	0.9674	0.9674	0.7368	0.7368	0.2631
NC	10	28					

Table 5.14. KDD CUP 99 dataset - independent ensemble method by combining LOF and RBDA algorithms with the average score: precision and recall for positive and negative class, false positive rate and confusion matrix.

ENSEMBLE: LOF-KNN-RADA - COMBINATION: MAXIMIZATION						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	288	19	0.9381	0.9381	0.5	0.5	0.5
NC	19	19					

Table 5.15. KDD CUP 99 dataset - independent ensemble method by combining LOF, KNN and RADA algorithms with the maximum score: precision and recall for positive and negative class, false positive rate and confusion matrix.

class increase by going from 0.6444 to 0.7567 and from 0.9478 to 0.9706 respectively. The false positive rate increases by going from 0.2368 to 0.2631. Moreover, looking also at the confusion matrix we can see that the algorithm RBDA correctly classifies 291 inliers and 29 outliers, while the independent ensemble method correctly classifies 298 inliers and 28 outliers, therefore we lose an outlier but we earn 7 inliers. Then, compared to the RBDA algorithm, we can conclude that there is an improvement but it's not as obvious as what we get with COF algorithm. All these analyses leads us to conclude that, for the first dataset, we were able to build independent ensemble methods that identify more accurately inliers and outliers. The same considerations can be made for the other two examples of independent ensemble methods reported in Tables 5.14 and 5.15 and obtained by combining LOF and RBDA with average score technique and by combining LOF, KNN

and RADA with maximum score technique, respectively.

5.4 Results sequential ensemble methods: KDD CUP 99 dataset

The sequential ensemble methods implemented are single-layer type, in particular they are the "*sequential-1 algorithm*" (4.4) and the "*sequential-2 algorithm* " (4.5) described in chapter 4. For both methods, β, i.e. the fraction of the original dataset D that the first algorithm of the sequence regards more abnormal, was set to 0.3. Therefore, $D_\beta = 0.3 \cdot |D|$ is the dataset used in the second step of method to calculate the anomaly score of all observations. In the *sequential-2 algorithm* is considered a percentage $\gamma = 0.4$ from D_β to obtain a sub-sample of D_β which is used, by the second algorithm of the sequence, to calculate the anomaly scores of all observations of the entire dataset. This process is repeated $T = 5$ times.

In the first example of sequential ensemble method (5.16) we implemented the *sequential-1 algorithm*, using as the first algorithm RADA and as the second algorithm COF. We can notice that the precision improves and it is 0.9640 and 0.6923 for positive and negative class, respectively. Instead, in the RADA algorithm the precision is 0.9066 for positive class and 0.2222 for negative class. In COF algorithm the precision is 0.9161 and 0.2741 for positive and negative class, respectively. The recall in this sequential ensemble method is 0.9609 for positive class and 0.7105 for negative class. Also in this case the recall is better than that obtained with the single algorithms. In fact, in the RADA we have 0.8859 and 0.2631 for positive and negative class, respectively. In the COF we have 0.8892 and 0.3421 for positive and negative class, respectively. The false positive rate decreases by going from 0.6578, for COF algorithm, and 0.7368, for the RADA algorithm to 0.2894. Then, the ensemble sequential method obtained by combining RADA, as first algorithm, and COF, as second algorithm, returns a better performance.

As for the previous case, also for the RADA-COF sequential ensemble method we report, in the Table 5.17, the mean and the standard deviation for all features and for both classes (inliers and outliers). Subsequently, we analyzed anomalous observations that were not correctly identified, i.e. false positives. Also for these observations we calculated the mean and standard deviation for all the features whose values are shown in the Table 5.18.

We noticed that 3 features "*Count*", "*diff_srv_rate*" and "*srv_doff_host_rate*", characterized by different behavior, could help to correctly classify these observations.

For these features, as we can see from the histograms shown in the Figure 5.4, the inliers assume certain values only in correspondence with these false positives, while all

SEQUENTIAL ENSEMBLE: RADA-COF - METHODS 1						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	295	11	0.9640	0.9609	0.6923	0.7105	0.2894
NC	12	27					

Table 5.16. KDD CUP 99 dataset - sequential-1 ensemble method: the first algorithm RADA and the second algorithm COF. $\beta = 0.3$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

	Mean_Inliers	**Std_Inliers**	**Mean_Outliers**	**Std_Outliers**
duration	99.467320	701.314406	0.923077	3.615643
src_bytes	574.163399	3181.042309	979.000000	581.386636
dst_bytes	2155.990196	4768.174613	9900.948718	44606.657607
hot	0.006536	0.114332	0.000000	0.000000
num_compromised	0.003268	0.057166	0.000000	0.000000
num_root	0.003268	0.057166	0.000000	0.000000
num_file_creations	0.000000	0.000000	0.0256641	0.160128
num_access_files	0.000000	0.000000	0.051282	0.223456
Count	15.029412	46.288856	354.153846	238.064182
srv_count	13.23026	36.509353	354.717949	237.223629
serror_rate	1.088235	9.925066	0.000000	0.000000
srv_serror_rate	1.035948	9.873728	0.000000	0.000000
rerror_rate	6.944444	25.328533	0.000000	0.000000
srv_rerror_rate	6.879085	25.320537	0.000000	0.000000
diff_srv_rate	97.209150	15.186989	94.435897	16.844169
srv_doff_host_rate	0.800654	6.051376	9.410256	28.578348
dst_host_count	11.532680	25.808257	13.333333	33.820994
dst_host_srv_count	152.676471	102.159839	237.333333	47.198536
dst_host_same_srv_rate	202.839869	85.813725	220.538462	74.696173
dsr_host_same_srv_rate	83.813725	31.107976	88.230769	28.544021
dst_host_diff_srv_rate	3.545752	14.238698	4.128205	15.766686
dst_host_same_src_port_rate	11.274510	24.426064	71.948718	45.340785
dst_host_srv_diff_host_rate	2.071895	3.480352	0.076923	0.269953
dst_host_serror_rate	1.088235	9.879706	0.000000	0.000000
dst_host_srv_serror_rate	1.058824	9.867473	0.000000	0.000000
dst_host_rerror_rate	6.830065	24.571286	0.000000	0.000000
dst_host_srv_rerror_rate	6.803922	24.454725	0.000000	0.000000

Table 5.17. **KDD CUP 99 dataset- Sequential RADA-COF:** mean and deviation standard for each feature for each class (inliers and outliers).

the other inliers assume different and much more frequent values.

However, the presence of only these 3 features was not sufficient for the algorithm to classify them correctly. We can apply the same considerations to the other cases of sequential ensemble methods reported in the Tables 5.19, 5.20 and 5.21 obtained by combining as the first algorithm KNN and as the second algorithm COF, by combining as the first algorithm RADA and as the second algorithm LOF and by combining as the first algorithm RADA and as the second algorithm KNN, respectively.

Therefore, we showed that, for the first dataset, KDD CUP 99, we can to build both

	Mean	Std
duration	0.000000	0.000000
src_bytes	5005.454545	16429.490779
dst_bytes	755.818182	2506.765319
hot	0.181818	0.603023
num_compromised	0.090909	0.301511
num_root	0.000000	0.000000
num_file_creations	0.000000	0.000000
num_access_files	0.000000	0.000000
Count	182.363636	135.786798
srv_count	52.545455	142.568134
serror_rate	27.272727	46.709937
srv_serror_rate	27.272727	46.709937
rerror_rate	54.545455	52.223297
srv_rerror_rate	54.545455	52.223297
diff_srv_rate	32.181818	43.675664
srv_doff_host_rate	4.454545	2.910795
dst_host_count	0.000000	0.000000
dst_host_srv_count	255.000000	0.000000
dst_host_same_srv_rate	54.181818	99.455335
dsr_host_same_srv_rate	21.181818	39.043100
dst_host_diff_srv_rate	13.909091	28.686075
dst_host_same_src_port_rate	18.181818	40.451992
dst_host_srv_diff_host_rate	0.000000	0.000000
dst_host_serror_rate	27.363636	46.652487
dst_host_srv_serror_rate	27.363636	46.652487
dst_host_rerror_rate	54.909091	51.817863
dst_host_srv_rerror_rate	54.909091	51.817863

Table 5.18. **KDD CUP 99 dataset- Sequential COF-RADA :** mean and standard deviation of false positives.

SEQUENTIAL ENSEMBLE: KNN-COF - METHODS 2						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	299	18	0.9432	0.9739	0.7142	0.5263	0.4736
NC	8	20					

Table 5.19. KDD CUP 99 dataset - sequential-2 ensemble method: the first algorithm KNN and the second algorithm COF. $\beta = 0.3$, $\gamma = 0.4$ and $T = 5$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

sequential and independent ensemble methods, which classify more accurately inliers and outliers than individual anomaly detection algorithms.

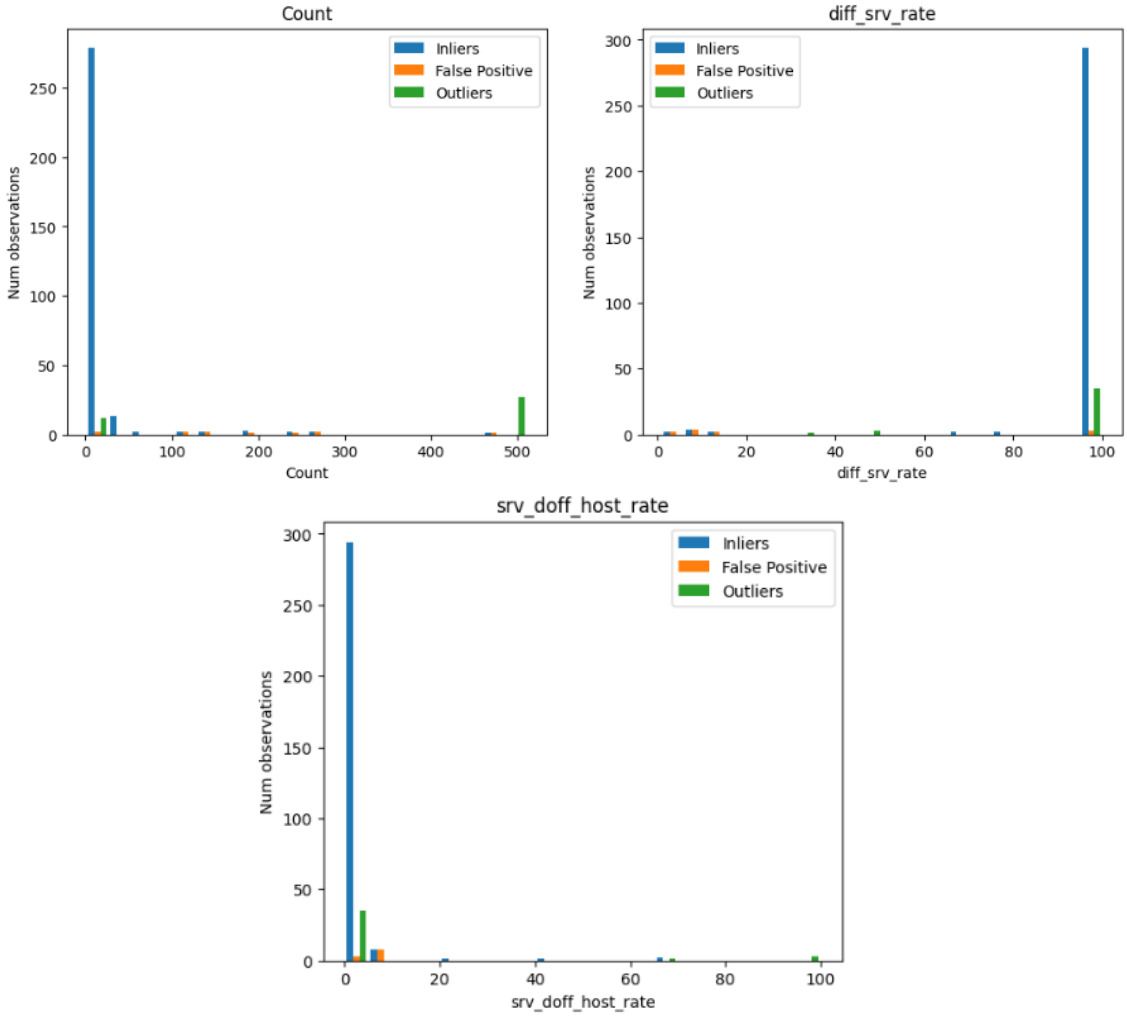

Figure 5.4. **KDD CUP 99 dataset- Sequential COF-RADA :** histograms of *Count*, *diff_srv_rate* and *srv_doff_host_rate* features (inliers in blue, false positives in orange and outliers in green).

5.5 Results anomaly detection algorithms: Simargl-2022 dataset

For the second dataset Simargl2022, the values obtained with the base learners are shown in the Tables 5.22, 5.23, 5.24, 5.25, 5.26 and 5.27. We set $k = 5$, $k = 20$, $k = 10$, $k = 20$, $k = 15$ and $k = 5$ for the LOF, COF, KNN, RBDA, RADA and LOCI, respectively. We observed that, for Simargl2022 dataset, the base algorithm RBDA achieves the best performance among all the individual base learners.

SEQUENTIAL ENSEMBLE: RADA-LOF - METHODS 2						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	302	29	0.9123	0.9837	0.6428	0.2368	0.7631
NC	5	9					

Table 5.20. KDD CUP 99 dataset - sequential-2 ensemble method: the first algorithm RADA and the second algorithm LOF. $\beta = 0.3$, $\gamma = 0.4$ and $T = 5$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

SEQUENTIAL ENSEMBLE: RADA-KNN - METHODS 1						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	303	29	0.9126	0.9869	0.6923	0.2368	0.7631
NC	4	9					

Table 5.21. KDD CUP 99 dataset - sequential-1 ensemble method: the first algorithm RADA and the second algorithm KNN. $\beta = 0.3$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

5.6 Results independent ensemble methods: Simargl-2022 dataset

Also for the second dataset, Simargl2022, we used to assemble the base learners and to build the independent ensemble methods the maximum score method, the score averaging method and the majority vote. We have considered all the possible combinations that can be obtained with the six basic methods used, but we show only just some of these.

In the first example of independent ensemble methods (5.28) we combined, with the score averaging method, LOF algorithm and RBDA algorithm. We can see that the performance of independent ensemble method (LOF, RBDA) is better than those obtained using one of anomaly detection algorithm with which it was built. In particular, we obtain a precision, both of positive and of negative class, greater than that obtained with the single algorithms, by going from 0.9046 to 0.9739 and from 0.2195 a 0.7894, respectively, compared with the LOF algorithm, and by going from 0.9733 to 0.9739 and from 0.6666 a 0.7894, respectively, compared with the RBDA algorithm. The recall also improves, going from 0.8957 to 0.9739 for positive class and from 0.2368 to 0.7894 for negative class, compared with the LOF algorithm, and going from 0.9511 to 0.9739 for positive class, while the recall of negative class is the same, compared with the RBDA algorithm.

The false positive rate is equal to 0.2105 and therefore decreases with respect to the LOF algorithm, whose value is 0.7631, while with respect to the RBDA algorithm, the

LOF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	275	29	0.9046	0.8957	0.2195	0.2368	0.7631
NC	32	9					

Table 5.22. Simargl2022 dataset - LOF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 5$.

COF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	268	35	0.8844	0.8729	0.0714	0.0789	0.9210
NC	39	3					

Table 5.23. Simargl2022 dataset - COF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 20$.

KNN						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	259	37	0.875	0.8436	0.0204	0.0263	0.9736
NC	48	1					

Table 5.24. Simargl2022 dataset - KNN algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 10$.

RBDA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	292	8	0.9733	0.9511	0.6666	0.7894	0.2105
NC	15	30					

Table 5.25. Simargl2022 dataset - RBDA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 20$.

value is the same. Therefore, the independent ensemble method constructed by combining LOF and RBDA, with the score averaging method, allows to classify better inliers and outliers than individual anomaly detection algorithms. As for the experiments conducted on the first dataset, we analyzed the features both for the observations classified, by the LOF-RBDA independent ensemble method, as inliers and for those classified as outliers. Then, we computed the mean and standard deviation for both inliers and outliers. The

RADA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	289	11	0.9633	0.9413	0.6	0.7105	0.2894
NC	18	27					

Table 5.26. Simargl2022 dataset - RADA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 15$.

LOCI						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	296	35	0.8942	0.9641	0.2142	0	0.9210
NC	11	3					

Table 5.27. Simargl2022 dataset - LOCI algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 5$.

ENSEMBLE: LOF-RBDA - COMBINATION: AVERAGE							
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	299	8	0.9739	0.9739	0.7894	0.7894	0.2105
NC	8	30					

Table 5.28. Simargl2022 dataset - independent ensemble method by combining LOF and RADA algorithms with the score averaging: precision and recall for positive and negative class, false positive rate and confusion matrix.

values obtained, for each feature and for each class, are shown in the Table 5.29. We can see that for some features, such as "*LAST_SWITCHED*", the values obtained are different between the two classes, while for other features, such as "*TCP_WIN_MAX_OUT*" and "*DST_TOS*", anomalies take on a single value.

Also, we noticed that the second dataset, Simargl2022, has, for the "*L4_SRC_PORT*" feature and for the "*TCP_FLAGS*" feature, the histograms report in the Figure 5.5 and compared to the other features they had a slightly more particular behavior. For this reason we wondered how some values, quite but not too frequent, were classified by the independent ensemble method LOF-RBDA. We observed that most of these values are present in both classes. In particular, the values assumed by observations identified as outliers are also values assumed by observations identified as inliers. It is possible to see this in the histograms of the two features mentioned above in the Figure 5.6. We tried to analyze why the method identified 8 observations as inliers but in reality they were outliers.

	Mean_Inliers	Std_Inliers	Mean_Outliers	Std_Outliers
L4_SRC_PORT	4.197895e+04	199229.389003	3.355911e+04	2.534211e+04
L4_DST_PORT	5.579156+03	15301.459584	2.044211e+03	8.210999e+03
FIRST_SWITCHED	1.647521e+09	11730.525810	1.647535e+09	3.926299e+04
FLOW_DURATION_MILLISECONDS	6.130423e+03	22122.924448	1.932447e+03	6.138978e+03
LAST_SWITCHED	1.647521e+09	11729.879216	1.647521e+09	3.926215e+04
PROTOCOL	1.070358e+01	5.844395	6.000000e+00	4.876862e+00
TCP_FLAGS	1.056352e+01	11.654267	3.736842e+00	5.976007e+00
TCP_WIN_MAX_IN	1.352266e+04	25653.809977	5.928421e+02	5.123640e+02
TCP_WIN_MAX_OUT	1.126744e+04	23694.658596	0.000000e+00	0.000000e+00
TCP_WIN_MIN_IN	1.351426e+04	25640.297191	5.928421e+02	5.123640e+02
TCP_WIN_MIN_OUT	1.124597e+04	23698.374503	0.000000e+00	0.000000e+00
TCP_WIN_MSS_IN	3.042215e+02	593.762350	5.763158e+02	7.232189e+02
TCP_WIN_SCALE_IN	1.394137e+00	2.993080	0.000000e+00	0.000000e+00
TCP_WIN_SCALE_OUT	1.267101e+00	2.932853	0.000000e+00	0.000000e+00
SRC_TOS	1.012378e+01	39.765624	3.578947e+00	1.111277e+01
DST_TOS	1.035831e+01	39.055708	0.000000e+00	0.000000e+00
TOTAL_FLOWS_EXP	3.575057e+08	631650.500723	3.578255e+08	3.109215e+06
MIN_IP_PKT_LEN	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
MAX_IP_PKT_LEN	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
TOTAL_PKTS_EXP	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
TOTAL_BYTES_EXP	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
IN_BYTES	2.065290e+03	12833.252559	1.597895e+02	5.900347e+02
IN_PKTS	1.247557e+01	121.777770	1.578947e+00	1.535729e+00
OUT_BYTES	2.286218e+04	352124.743829	1.333158e+02	7.672970e+02
OUT_PKTS	1.944625e+01	241.261458	2.894737e+01	1.313298e+00

Table 5.29. **Simargl2022 - Independent LOF-RBDA:** mean and variance for each feature for each class (inliers and outliers).

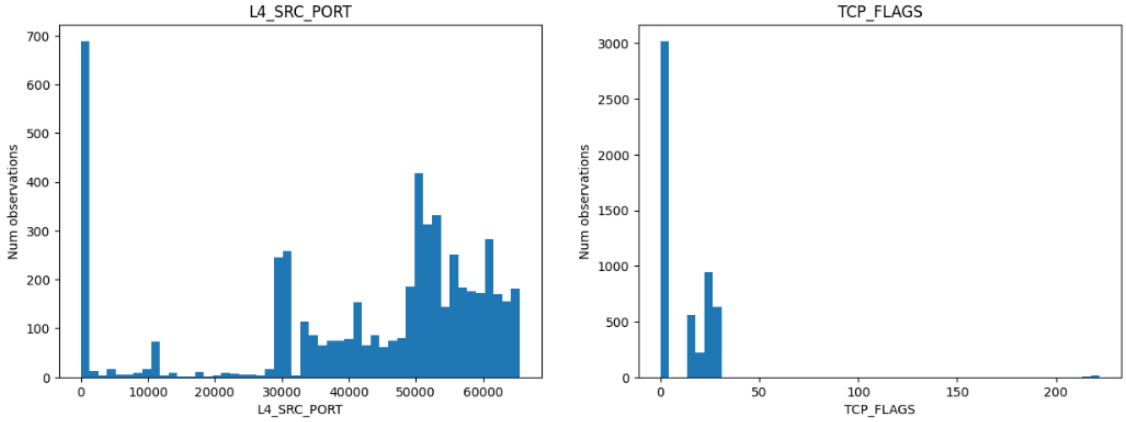

Figure 5.5. **Simargl2022 dataset:** histogram *L4_SRC_PORT* feature and histogram *TCP_FLAGS* feature.

To do this, we first calculated the mean and standard deviation of these false positives, report in the Table 5.30. Then, we analyzed the individual features using histogram plots. Also in this case we noticed that most of the features assumed, in correspondence with these observations, values present in both classes. The only features that could help classify these observations correctly were "*FIRST_SWITCHED*", "*LAST_SWITCHED*"

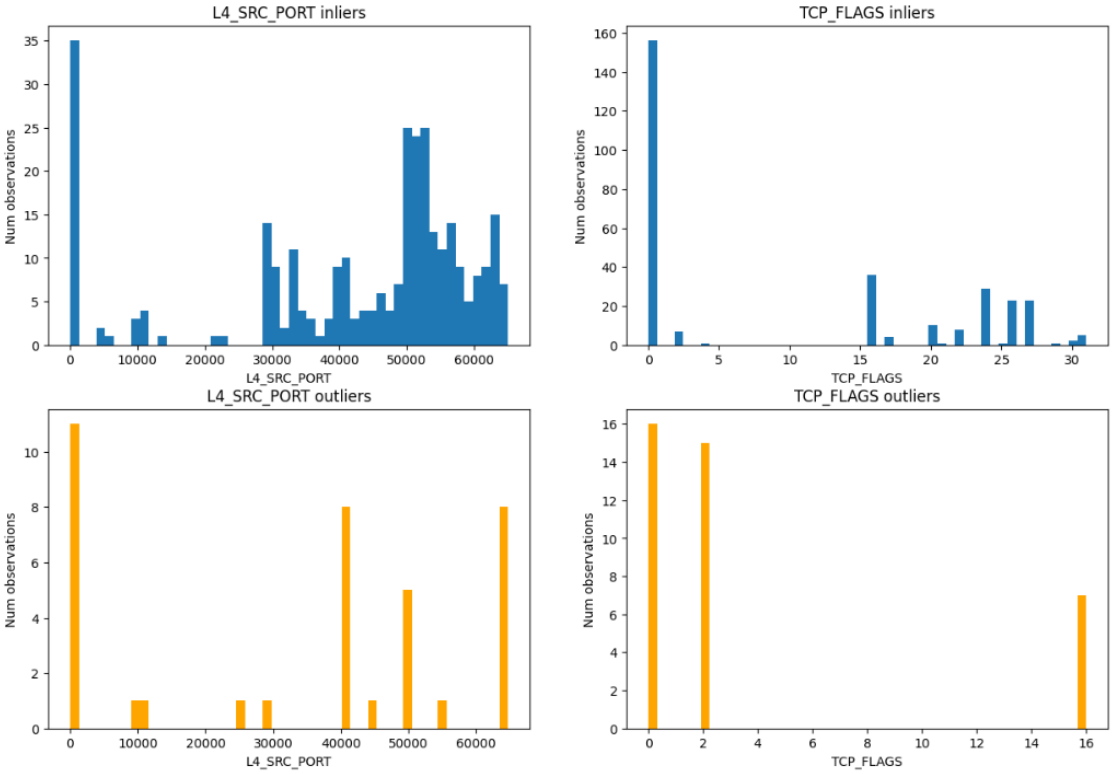

Figure 5.6. **Simargl2022 dataset - Independent LOF-RBDA:** histograms of *L4_SRC_PORT* and *TCP_FLAGS* features (inliers in blue, outliers in orange).

and "*TOTAL_FLOWS_EXP*". In fact, as we can see from the histograms shown in the Figure 5.7, the inliers assume specific values only in correspondence with these false positives. This could help to understand that the behavior of these observations differed from that of the inliers.

In the Table 5.31 we reported the results obtained by combining RBDA and RADA with majority vote technique, while in the Table 5.32 we reported the results obtained by combining three anomaly detection algorithms (LOCI, RBDA and RADA) using the majority vote technique.

5.7 Results sequential ensemble methods: Simargl-2022 dataset

The sequential ensemble methods implemented for dataset Simargl2022 are single-layer type and they are "*sequential-1 algorithm*" (4.4) and the "*sequential-2 algorithm* " (4.5).

	Mean	Std
L4_SRC_PORT	2.494238e+04	15394.768081
L4_DST_PORT	1.961250e+02	207.097932
FIRST_SWITCHED	1.647593e+09	154.381196
FLOW_DURATION_MILLISECONDS	2.303750e+03	6198.288652
LAST_SWITCHED	1.647593e+09	150.118049
PROTOCOL	4.750000e+00	2.314550
TCP_FLAGS	6.750000e+00	7.704359
TCP_WIN_MAX_IN	7.680000e+02	474.019891
TCP_WIN_MAX_OUT	0.000000e+00	0.000000e+00
TCP_WIN_MIN_IN	7.680000e+02	474.019891
TCP_WIN_MIN_OUT	0.000000e+00	0.000000e+00
TCP_WIN_MSS_IN	5.475000e+02	755.621787
TCP_WIN_SCALE_IN	0.000000e+00	0.000000e+00
TCP_WIN_SCALE_OUT	0.000000e+00	0.000000e+00
SRC_TOS	0.000000e+00	0.000000e+00
DST_TOS	0.000000e+00	0.000000e+00
TOTAL_FLOWS_EXP	3.613570e+08	16063.315367
MIN_IP_PKT_LEN	0.000000e+00	0.000000e+00
MAX_IP_PKT_LEN	0.000000e+00	0.000000e+00
TOTAL_PKTS_EXP	0.000000e+00	0.000000e+00
TOTAL_BYTES_EXP	0.000000e+00	0.000000e+00
IN_BYTES	5.550000e+01	26.484497
IN_PKTS	1.500000e+00	1.069045
OUT_BYTES	0.000000e+00	0.000000e+00
OUT_PKTS	0.000000e+00	0.000000e+00

Table 5.30. **Simargl2022 dataset - Independent LOF-RBDA:** mean and standard deviation of false positives.

ENSEMBLE: RBDA-RADA - COMBINATION: MAJORITY VOTE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	307	11	0.9654	1.0	1.0	0.7105	0.2894
NC	0	27					

Table 5.31. Simargl2022 dataset - independent ensemble method by combining RBDA and RADA algorithms with the majority vote: precision and recall for positive and negative class, false positive rate and confusion matrix.

For both methods, β, i.e. the fraction of observations of the original dataset D more abnormal for the first algorithm, is always set to 0.3. In the *sequential-2 algorithm* is considered a percentage $\gamma = 0.4$ from D_β to obtain a sub-sample of D_β that is used in the second step of method to calculate the anomaly scores of all observation of the entire dataset. This process is repeated $T = 5$ times.

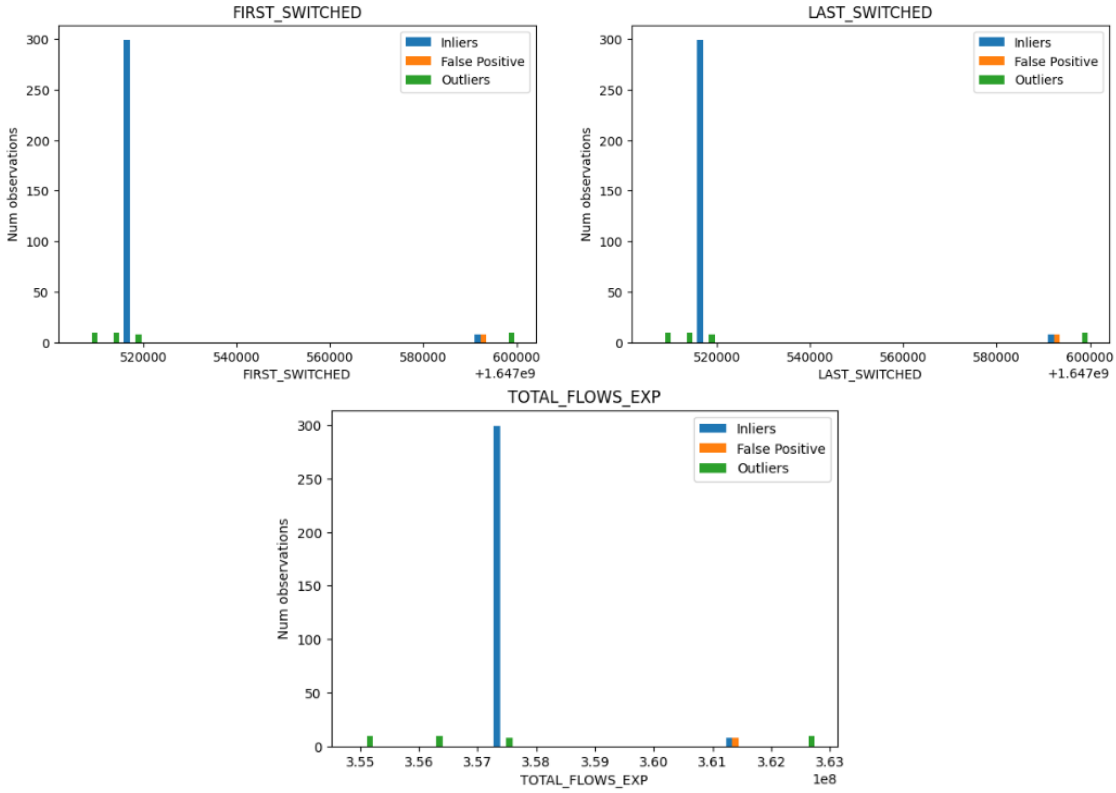

Figure 5.7. **Simargl2022 dataset- Sequential LOF-RBDA :** histograms of *FIRST_SWITCHED*, *LAST_SWITCHED* and *TOTAL_FLOWS_EXP* features (inliers in blue, false positives in orange and outliers in green).

ENSEMBLE: LOCI-RBDA-RADA - COMBINATION: MAJORITY VOTE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	303	11	0.9649	0.9869	0.8709	0.7105	0.2894
NC	4	27					

Table 5.32. Simargl2022 dataset - independent ensemble method by combining LOCI, RBDA and RADA algorithms with the majority vote: precision and recall for positive and negative class, false positive rate and confusion matrix.

In the first example of sequential ensemble method (5.33) we implemented the sequential algorithm 1, using as first algorithm LOF and as second algorithm KNN. We can notice that the precision improves and it is 0.9356 and 0.5294 for positive and negative class, respectively. Instead, in LOF algorithm the precision is 0.9046 for positive class and 0.2195 for negative class. In KNN algorithm precision is 0.875, for positive class, and

SEQUENTIAL ENSEMBLE: LOF- KNN - METHODS 1						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	291	20	0.9356	0.9478	0.5294	0.4736	0.5263
NC	16	18					

Table 5.33. Simargl2022 dataset - sequential-1 ensemble method: the first algorithm LOF and the second algorithm KNN. $\beta = 0.3$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

	Mean_Inliers	Std_Inliers	Mean_Outliers	Std_Outliers
L4_SRC_PORT	4.274014e+04	199220.298399	2.560588e+04	2.187475e+04
L4_DST_PORT	5.130260e+03	14394.273057	5.734412e+03	1.768464e+03
FIRST_SWITCHED	1.647518e+09	2501.109800	1.647560e+09	3.938919e+04
FLOW_DURATION_MILLISECONDS	4.701572e+03	1895.212090	1.450835e+04	3.369667e+04
LAST_SWITCHED	1.647518e+09	2501.915248	1.647560e+09	3.937483e+04
PROTOCOL	1.072990e+01	5.820795	5.205882e+00	4.409215e+00
TCP_FLAGS	1.013505e+01	11.529531	6.852941e+00	9.407065e+00
TCP_WIN_MAX_IN	1.265403e+04	25037.416316	7.017059e+03	1.886366e+04
TCP_WIN_MAX_OUT	1.099647e+04	23572.682546	1.153029e+03	5.119098e+03
TCP_WIN_MIN_IN	1.264575e+04	25023.454624	7.016941e+03	1.886370e+04
TCP_WIN_MIN_OUT	1.097529e+04	23576.117917	1.152853e+03	5.119066e+03
TCP_WIN_MSS_IN	3.144437e+02	601.036263	5.148235e+02	7.075595e+02
TCP_WIN_SCALE_IN	1.286174e+00	2.908015	8.235294e-01	2.328618e+00
TCP_WIN_SCALE_OUT	1.199357e+00	2.867864	4.705882e-01	1.910661e+00
SRC_TOS	4.360129e+00	23.563207	5.552941e+01	8.484004e+01
DST_TOS	8.385852+00	34.402630	1.682353e+01	5.540620e+01
TOTAL_FLOWS_EXP	3.572828e+08	490373.853693	3.599019e+08	2.452287e+06
MIN_IP_PKT_LEN	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
MAX_IP_PKT_LEN	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
TOTAL_PKTS_EXP	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
TOTAL_BYTES_EXP	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
IN_BYTES	1.262711e+03	8813.591634	7.276853e+03	2.771190e+04
IN_PKTS	5.016077e+00	11.565092	6.852941e+01	3.641898e+02
OUT_BYTES	2.675174e+03	18224.711919	1.821111e+05	1.057000e+06
OUT_PKTS	5.241158e+00	16.632160	1.279706e+02	7.235525e+02

Table 5.34. **Simargl2022 dataset - Sequential LOF-KNN:** mean and standard deviation for each feature for each class (inliers and outliers).

0.0204 for negative class, respectively. The recall in this sequential ensemble method is 0.9478 for positive class and 0.4736 for negative class. Also in this case the recall is better than that obtained with the single algorithms. In fact, we have in LOF 0.8957 and 0.2368 and we have in KNN 0.8436 and 0.0263 for the positive and the negative class, respectively. The false positive rate decreases by going from 0.7631, for LOF algorithm, and 0.9736, for the KNN algorithm, to 0.5263 for the sequential ensemble method. Therefore, the ensemble sequential algorithm, obtained by combining LOF, as first algorithm, and KNN, as second algorithm, returns a better performance.

As for the previous case, also for the LOF-KNN sequential ensemble method we report, in the Table 5.34, the mean and the standard deviation for all features and for both classes

	Mean	Std
L4_SRC_PORT	3.721415e+04	28874.559311
L4_DST_PORT	1.958500e+02	209.165831
FIRST_SWITCHED	1.647510e+09	4251.535345
FLOW_DURATION_MILLISECONDS	9.015000e+02	2944.806660
LAST_SWITCHED	1.647510e+09	4250.861478
PROTOCOL	4.800000e+00	3.750088
TCP_FLAGS	4.000000e+00	6.223893
TCP_WIN_MAX_IN	6.144000e+02	514.687683
TCP_WIN_MAX_OUT	0.000000e+00	0.000000e+00
TCP_WIN_MIN_IN	6.144000e+02	514.687683
TCP_WIN_MIN_OUT	0.000000e+00	0.000000e+00
TCP_WIN_MSS_IN	5.840000e+02	733.832047
TCP_WIN_SCALE_IN	0.000000e+00	0.000000e+00
TCP_WIN_SCALE_OUT	0.000000e+00	0.000000e+00
SRC_TOS	0.000000e+00	0.000000e+00
DST_TOS	0.000000e+00	0.000000e+00
TOTAL_FLOWS_EXP	3.556613e+08	784101.699549
MIN_IP_PKT_LEN	0.000000e+00	0.000000e+00
MAX_IP_PKT_LEN	0.000000e+00	0.000000e+00
TOTAL_PKTS_EXP	0.000000e+00	0.000000e+00
TOTAL_BYTES_EXP	0.000000e+00	0.000000e+00
IN_BYTES	5.325000e+01	18.367233
IN_PKTS	1.350000e+00	0.587143
OUT_BYTES	2.045000+01	91.455180
OUT_PKTS	5.000000e-02	0.223607

Table 5.35. **Simargl2022 dataset- Sequential LOF-KNN:** mean and standard deviation of false positives.

(inliers and outliers). Subsequently, we analyzed anomalous observations that were not correctly identified, i.e. false positives. Also for these observations we calculated the mean and standard deviation for all the features whose values are shown in the Table 5.35.

We noticed that three features, "*FIRST_SWITCHED*", "*LAST_SWITCHED*" and "*TOTAL_FLOWS_EXP*", could help to correctly classify these observations. For these features, as we can see from the histograms shown in the Figure 5.8, the inliers assume certain values only in correspondence with these false positives, while all the other inliers assume different and much more frequent values. However, the presence of only these 3 features was not sufficient for the algorithm to classify them correctly.

We can apply the same considerations to the other cases of sequential ensemble methods reported in the Tables 5.36 and 5.37 obtained by combining as the first algorithm KNN and as the second algorithm LOF and by combining as the first algorithm RBDA and as the second algorithm LOF, respectively.

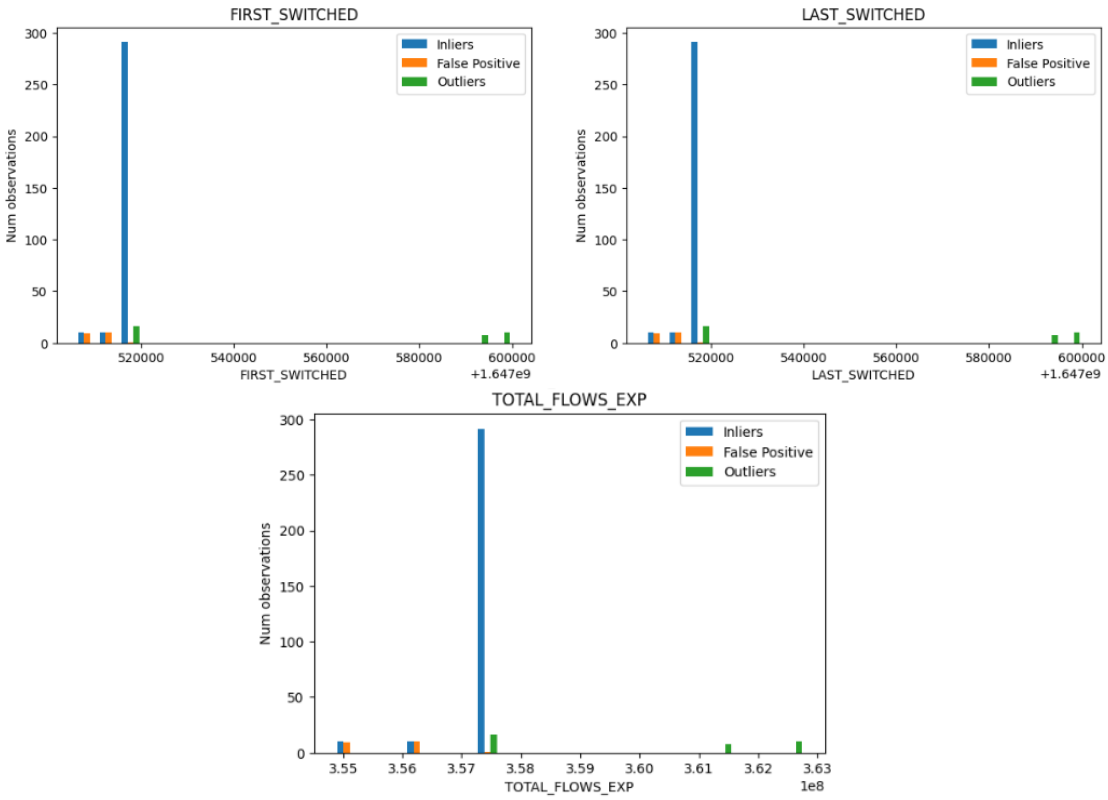

Figure 5.8. **Simargl2022 dataset- Sequential LOF-KNN:** histograms of *FIRST_SWITCHED*, *LAST_SWITCHED* and *TOTAL_FLOWS_EXP* features (inliers in blue, false positives in orange and outliers in green).

SEQUENTIAL ENSEMBLE: KNN-LOF - METHODS 2						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	301	23	0.9290	0.9804	0.7142	0.3947	0.6052
NC	6	15					

Table 5.36. Simargl2022 dataset - sequential-2 ensemble method: the first algorithm KNN and the second algorithm LOF. $\beta = 0.3$, $\gamma = 0.4$ and $T = 5$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

5.8 Results anomaly detection algorithms: IoT Network Intrusion dataset

For the third dataset, IoT Network Intrusion, the values obtained with the base learners are shown in the Tables 5.38, 5.39, 5.40, 5.41, 5.42 and 5.43.

SEQUENTIAL ENSEMBLE: RBDA-LOF - METHODS 1						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	307	12	0.9623	1.0	1.0	0.6842	0.3115
NC	0	26					

Table 5.37. Simargl2022 dataset - sequential-1 ensemble method: the first algorithm RBDA and the second algorithm LOF. $\beta = 0.3$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

LOF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	277	30	0.9022	0.9022	0.2105	0.2105	0.7894
NC	30	8					

Table 5.38. IoT Network Intrusion dataset - LOF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 20$.

COF						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	272	26	0.9127	0.8859	0.2553	0.3157	0.6842
NC	35	12					

Table 5.39. IoT Network Intrusion dataset - COF algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

KNN						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	287	28	0.9111	0.9348	0.3333	0.2631	0.7368
NC	20	10					

Table 5.40. IoT Network Intrusion dataset - KNN algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

We set $k = 20$, $k = 25$, $k = 25$, $k = 10$, $k = 25$ and $k = 25$ for the LOF, COF, KNN, RBDA, RADA and LOCI, respectively. We observed that, for IoT Network Intrusion dataset, the base algorithm KNN achieves the best performance among all the individual base algorithms.

RBDA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	270	36	0.8823	0.8794	0.0512	0.0526	0.9473
NC	37	2					

Table 5.41. IoT Network Intrusion dataset - RBDA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 10$.

RADA						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	275	25	0.9166	0.8957	0.2888	0.3421	0.6578
NC	32	13					

Table 5.42. IoT Network Intrusion dataset - RADA algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

LOCI						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	305	35	0.8970	0.9934	0.6	0.0789	0.9210
NC	2	3					

Table 5.43. IoT Network Intrusion dataset - LOCI algorithm: precision and recall for positive and negative class, false positive rate and confusion matrix. $k = 25$.

5.9 Results independent ensemble methods: IoT Network Intrusion dataset

Also for the third dataset, IoT Network Intrusion, we used the maximum score method, the score averaging method and the majority vote to assemble the base learners and to build the independent ensemble methods. We have considered all possible combinations that can be obtained with the 6 basic methods used, but we show only just some of these.

For this dataset it is less evident the improvement of the performances obtained with the methods of independent ensemble compared with the single algorithms of anomaly detection. For example, in the case (5.44), we combined, with the maximum score method, LOF and RADA algorithms. We obtain a precision of 0.9120 and 0.2894 for positive and negative class, respectively. In the LOF algorithm, the precision, for both classes, is lower (0.9022 for positive class and 0.2105 for negative class), while in the RADA algorithm we

ENSEMBLE: LOF-RADA - COMBINATION: MAXIMIZATION						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	280	27	0.9120	0.9120	0.2894	0.2894	0.7105
NC	27	11					

Table 5.44. IoT Network Intrusion dataset - independent ensemble method by combining LOF and RADA algorithms with the maximum score: precision and recall for positive and negative class, false positive rate and confusion matrix.

ENSEMBLE: KNN-RADA - COMBINATION: MAJORITY VOTE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	293	30	0.9071	0.9543	0.3636	0.2105	0.7894
NC	14	8					

Table 5.45. IoT Network Intrusion dataset - independent ensemble method by combining KNN and RADA algorithms with the majority vote: precision and recall for positive and negative class, false positive rate and confusion matrix.

ENSEMBLE: LOF-COF - COMBINATION: MAJORITY VOTE						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	294	31	0.9046	0.9576	0.35	0.1842	0.8157
NC	13	7					

Table 5.46. IoT Network Intrusion dataset - independent ensemble method by combining LOF and COF algorithms with the majority vote: precision and recall for positive and negative class, false positive rate and confusion matrix.

get very similar values (0.9166 and 0.2888 for positive and negative class, respectively). The recall, in the independent ensemble method, is 0.9120 and 0.2894 for positive and negative class, respectively. Also in this case the recall values are higher than those obtained with the LOF algorithm (0.9022 and 0.2105 for positive and negative class), while in RADA algorithm the recall is greater for negative class (0.3421), but is lower for positive class (0.8957). The false positive rate is equal to 0.7105 and therefore decreases compared with the LOF algorithm, whose value is 0.7894, while increases compared with the RADA algorithm, whose value is 0.6578. Since in this case the improvement is not so evident we also analyze the confusion matrix. We note that the algorithm RADA correctly classifies 275 inliers and 13 outliers, while the method of independent ensemble correctly classifies 280 inliers and 11 outliers, then we lose 2 outlier but we earn 5 inliers. Therefore, compared

with the RADA algorithm, we can conclude that there is a small improvement, but it is not so obvious unlike those obtained with previous datasets. The same considerations can be made for the other two examples of independent ensemble methods reported in Tables 5.45 and 5.46 and obtained by combining KNN and RADA with majority vote technique and by combining LOF and COF with majority vote technique, respectively.

5.10 Results sequential ensemble methods: IoT Network Intrusion dataset

The sequential ensemble methods implemented for IoT Network Intrusion dataset are always single-layer type and, like the previous cases, they are "*sequential-1 algorithm*" (4.4) and the "*sequential-2 algorithm* " (4.5). For both methods, β, i.e. the fraction of observations of the original dataset D more abnormal for the first algorithm, is always set to 0.3. In the *sequential-2 algorithm* is considered a percentage $\gamma = 0.4$ from D_β to obtain a sub-sample of D_β, used in the second step of method to calculate the anomaly scores of all observation of the entire dataset. This process is repeated $T = 5$ times.

In the first case of sequential ensemble method (5.47) we implemented the sequential algorithm 2, using as first algorithm LOF and as second algorithm COF. We can notice that precision improves and it is 0.9266 and 0.3555 for positive and negative class, respectively. Instead, in LOF algorithm the precision is 0.9022 for positive class and 0.2105 for negative class. In COF algorithm precision is 0.9127 and 0.2553 for positive and negative class, respectively. The recall in this sequential ensemble method is 0.9055 for positive class and 0.4210 for negative class. Also in this case the recall is better than that obtained with the single algorithms, in fact we have in LOF 0.9022 and 0.2105 and in COF 0.8859 and 0.3157 for positive and negative class, respectively. The false positive rate decreases by going from 0.7894, for LOF algorithm, and 0.6842, for the COF algorithm, to 0.5789. Therefore, the ensemble sequential method, obtained by combining LOF and COF, returns better performance.

In the second case (5.48) we implemented the sequential algorithm 1, using as first algorithm RBDA and as second algorithm COF. The precision improves and it is 0.9395 and 0.4255 for positive and negative class, respectively. Instead, in RBDA algorithm the precision is 0.8823 for positive class and 0.0512 for negative class. In COF algorithm the precision is 0.9127 and 0.2553 for positive and negative class, respectively. The recall of ensemble method is 0.9120 for positive class and 0.5263 for negative class. Also in this case the recall is better than that obtained with the single algorithms, in fact we have in RBDA 0.8794 and 0.0526 and in COF 0.8859 and 0.3157 for positive and negative class, respectively.

SEQUENTIAL ENSEMBLE: LOF-COF - METHODS 2						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	278	22	0.9266	0.9055	0.3555	0.4210	0.5789
NC	29	16					

Table 5.47. IoT Network Intrusion dataset - sequential-2 ensemble method: the first algorithm LOF and the second algorithm COF. $\beta = 0.3$, $\gamma = 0.4$ and $T = 5$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

SEQUENTIAL ENSEMBLE: RBDA-COF - METHODS 1						
Confusion Matrix			Precision PC	Recall PC	Precision NC	Recall NC	FPR
	PC	NC					
PC	280	18	0.9395	0.9120	0.4255	0.5263	0.4736
NC	27	20					

Table 5.48. IoT Network Intrusion dataset - sequential-1 ensemble method: the first algorithm RBDA and the second algorithm COF. $\beta = 0.3$, $\gamma = 0.4$ and $T = 5$. Precision and recall for positive and negative class, false positive rate and confusion matrix.

The false positive rate decreases by going from 0.9473, for RBDA algorithm, and 0.6842, for the COF algorithm, to 0.4736. Then, with the ensemble sequential method, obtained by combining RBDA and COF, we have better performance than those of the base learners with which it is built.

As for the experiments conducted on previous datasets, we analyzed the features both for the observations classified, by the RBDA-COF sequential ensemble method, as inliers and for those classified as outliers. Then, we computed the mean and standard deviation for both inliers and outliers. The values obtained, for each feature and for each class, are shown in the Table 5.49.

We can see that for some features, as "*Bwd_Pkt_Len_Max*" and "Subflow_Fwd_Byts", the values obtained for the two classes are different, while for other features, such as "*Down/Up_Ratio*" and "*Fwd_Header_Len*", the values are very similar. Also, we noticed that the third dataset, IoT Network Intrusion, has, for the "*Pkt_Len_Mean*" feature and for the "*Pkt_Size_Avg*" feature, the histograms report in the Figure 5.9 and compared to the other features they had a slightly more particular behavior. For this reason we wondered how values were classified by the sequential ensemble method RBDA-COF. We observed that the most frequent value for both features, i.e. 0, is present in both classes, while other values more frequent are present only in the inlier class.

Furthermore, as we can see from the histograms in the Figure 5.10, outliers are mainly concentrated between 0 and 50, for "*Pkt_Len_Mean*" feature, and between 0 and 100, for

	Mean_Inliers	Std_Inliers	Mean_Outliers	Std_Outliers
Src_Port	13138.4161	13899.1922	41819.0000	19225.3668
Dst_Port	44950.6812	13919.6625	14324.4893	15347.1981
Flow_Duration	320.0402	146.0371	331.6382	302.7305
Tot_Fwd_Pkts	1.2449	1.2487	1.7234	1.6511
Tot_Bwd_Pkts	1.6442	1.0021	1.1702	0.3798
TotLen_Fwd_Pkts	1332.3557	1483.4522	63.6808	135.1633
TotLen_Bwd_Pkts	1225.0335	992.7757	33.5531	97.8018
Fwd_Pkt_Len_Max	810.9530	682.8264	54.0638	132.2028
Fwd_Pkt_Len_Min	512.6845	657.0177	54.0638	132.2028
Fwd_Pkt_Len_Mean	690.0061	618.2730	54.0638	132.2028
Fwd_Pkt_Len_Std	186.8079	354.0427	0.0000	0.0000
Bwd_Pkt_Len_Max	1048.0704	600.2487	33.1276	97.7277
Bwd_Pkt_Len_Min	965.5033	641.0247	32.1489	97.8530
Bwd_Pkt_Len_Mean	1005.9745	601.8574	32.1489	97.7617
Bwd_Pkt_Len_Std	54.3013	214.3828	0.6920	3.3475
Flow_Pkts/s	11290.0568	6113.4917	27282.8664	59978.5615
Flow_IAT_Mean	195.2819	117.7501	264.0930	316.6240
Flow_IAT_Std	41.7556	60.8529	18.6176	45.1608
Flow_IAT_Max	232.5604	123.3404	285.0638	311.7255
Flow_IAT_Min	165.8053	128.3912	252.8297	322.4592
Flow_IAT_Tot	87.3959	128.3338	70.8936	128.7559
Fwd_IAT_Mean	62.1381	92.6493	36.1697	59.0200
Fwd_IAT_Std	7.8903	23.2014	18.9092	55.2197
Fwd_IAT_Max	68.8288	99.8002	50.1914	93.8761
Fwd_IAT_Min	56.5503	89.1534	22.9148	34.3497
Bwd_IAT_Tot	124.6711	188.8637	100.7659	322.9169
Bwd_IAT_Mean	85.6669	144.5583	100.7659	322.9169
Bwd_IAT_Std	16.6363	47.1440	0.0000	0.0000
Bwd_IAT_Max	101.9026	161.0703	100.7659	322.9169
Bwd_IAT_Min	74.7953	140.6629	100.7659	322.9169
Fwd_Header_Len	39.4765	40.0672	42.7234	44.7610
Bwd_Header_Len	52.1744	32.2836	29.7872	15.3522
Fwd_Pkts/s	4430.5068	4097.2335	16922.3049	40656.5121
Bwd_Pkts/s	6859.5500	6007.9881	10360.5614	21228.5889
Pkt_Len_Min	587.2483	670.9253	32.1489	97.8530
Pkt_Len_Max	1059.7181	594.7591	55.4680	131.7949
Pkt_Len_Mean	926.9939	557.6641	39.0709	99.6490
Pkt_Len_Std	235.9366	319.2657	12.6733	52.0268
Pkt_Len_Var	157254.6922	225698.0416	2809.8156	13066.0836
Down/Up_Ratio	0.2818	0.5135	0.4680	0.5457
Pkt_Size_Avg	1282.3194	787.0066	57.2529	149.2503
Fwd_Seg_Size_Avg	690.0061	618.2730	54.0638	132.2028
Bwd_Seg_Size_Avg	1005.9745	601.8574	32.6382	97.7617
Subflow_Fwd_Pkts	1.2449	1.2487	1.7234	1.6511
Subflow_Fwd_Byts	1332.3557	1483.4522	63.6808	135.1633
Subflow_Bwd_Pkts	1.6442	1.0021	1.1702	0.3798
Subflow_Bwd_Byts	1225.0335	992.7757	33.5531	97.8018
Init_Bwd_Win_Byts	10607.8590	13886.3328	18373.5744	15261.9905
Fwd_Act_Data_Pkts	1.2248	1.2498	0.6808	1.3847
Active_Mean	0.0000	0.0000	0.0000	0.0000
Active_Std	0.0000	0.0000	0.0000	0.0000
Active_Max	0.0000	0.0000	0.0000	0.0000
Active_Min	0.0000	0.0000	0.0000	0.0000
Idle_Mean	195.2819	117.7501	246.3483	318.5221
Idle_Std	41.7556	60.8529	18.6176	45.1608

Table 5.49. **IoT Network Intrusion dataset- Sequential RBDA-COF :** mean and standard deviation of for each feature for each class (inliers and outliers).

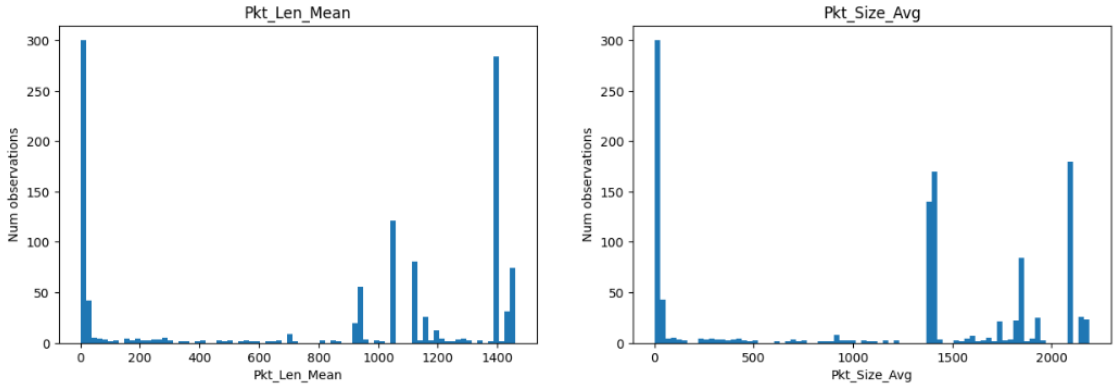

Figure 5.9. **IoT Network Intrusion dataset:** histogram *Pkt_Len_Mean* feature and histogram *Pkt_Size_Avg* feature.

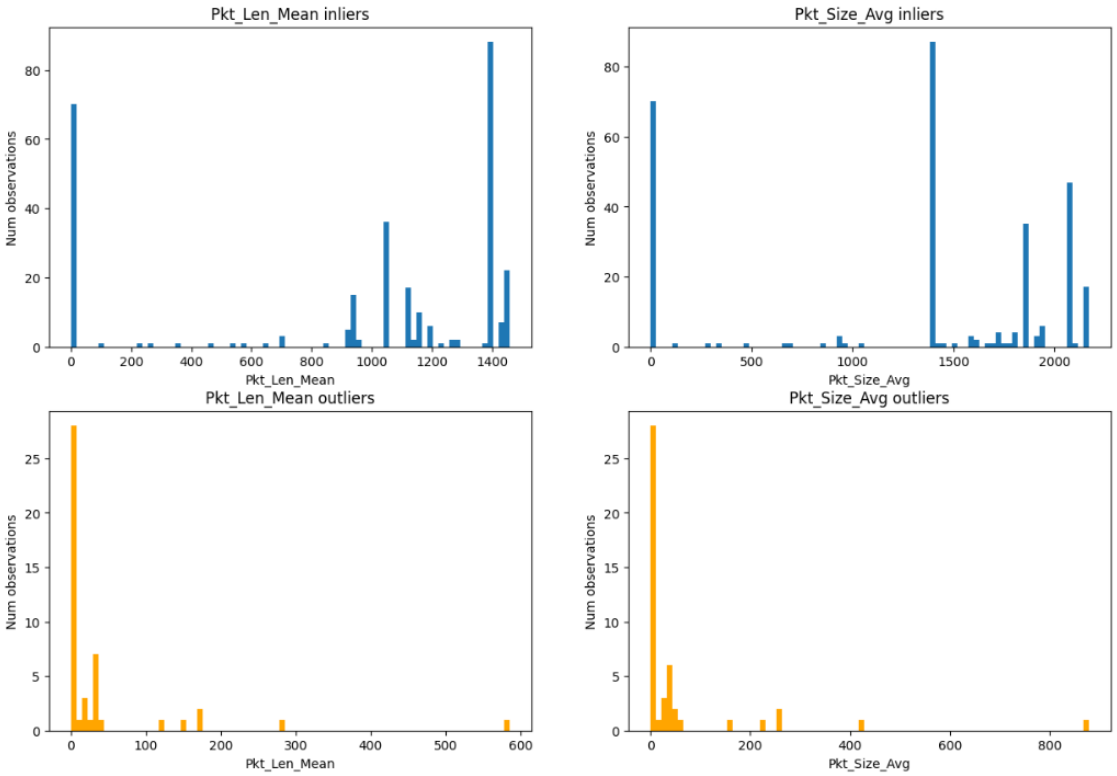

Figure 5.10. **IoT Network Intrusion dataset - Sequential RBDA-COF:** histograms of *Pkt_Len_Mean* and *Pkt_Size_Avg* features (inliers in blue, outliers in orange).

"*Pkt_Size_Avg*" feature. We tried to analyze why the method identified 18 observations

	Mean	Std		Mean	Std
Src_Port	32579.3333	25846.9464	**Bwd_IAT_Max**	36.6111	49.9259
Dst_Port	23576.7222	20723.1789	**Bwd_IAT_Min**	35.7222	49.0927
Flow_Duration	116.2777	41.5023	**Fwd_Header_Len**	16.6666	18.5218
Tot_Fwd_Pkts	0.6666	0.5940	**Bwd_Header_Len**	41.5555	24.8151
Tot_Bwd_Pkts	1.4444	0.6156	**Fwd_Pkts/s**	6009.3453	5694.0766
TotLen_Fwd_Pkts	871.3333	826.2097	**Bwd_Pkts/s**	14170.7434	7680.7853
TotLen_Bwd_Pkts	1969.6666	694.5346	**Pkt_Len_Min**	1265.2222	389.3626
Fwd_Pkt_Len_Max	794.2222	691.8322	**Pkt_Len_Max**	1421.5555	26.9928
Fwd_Pkt_Len_Min	794.2222	691.8322	**Pkt_Len_Mean**	1375.7314	112.2899
Fwd_Pkt_Len_Mean	794.2222	691.8322	**Pkt_Len_Std**	84.4234	205.0702
Fwd_Pkt_Len_Std	0.0000	0.0000	**Pkt_Len_Var**	46844.7962	125197.6511
Bwd_Pkt_Len_Max	1421.5555	26.9928	**Down/Up_Ratio**	0.5555	0.5113
Bwd_Pkt_Len_Min	1346.1111	329.4588	**Pkt_Size_Avg**	2041.0370	205.8302
Bwd_Pkt_Len_Mean	1396.4074	117.8954	**Fwd_Seg_Size_Avg**	794.2222	691.8322
Bwd_Pkt_Len_Std	43.5578	184.8003	**Bwd_Seg_Size_Avg**	1396.4074	117.8954
Flow_Pkts/s	20180.0888	6419.6223	**Subflow_Fwd_Pkts**	0.6666	0.5940
Flow_IAT_Mean	107.8055	40.9343	**Subflow_Fwd_Byts**	871.3333	826.2097
Flow_IAT_Std	0.6678	2.6620	**Subflow_Bwd_Pkts**	1.4444	0.6156
Flow_IAT_Max	108.2777	40.6314	**Subflow_Bwd_Byts**	1969.6666	694.5346
Flow_IAT_Min	107.3333	41.3208	**Init_Bwd_Win_Byts**	809.0555	861.0681
Flow_IAT_Tot	4.0000	16.9705	**Fwd_Act_Data_Pkts**	0.6666	0.5940
Fwd_IAT_Mean	4.0000	16.9705	**Active_Mean**	0.0000	0.0000
Fwd_IAT_Std	0.0000	0.0000	**Active_Std**	0.0000	0.0000
Fwd_IAT_Max	4.0000	16.9705	**Active_Max**	0.0000	0.0000
Fwd_IAT_Min	4.0000	16.9705	**Active_Min**	0.0000	0.0000
Bwd_IAT_Tot	40.6111	56.7088	**Idle_Mean**	107.8055	40.9343
Bwd_IAT_Mean	36.1666	49.4751	**Idle_Std**	0.6678	2.6620
Bwd_IAT_Std	0.6285	2.6666			

Table 5.50. **IoT Network Intrusion dataset- Sequential RBDA-COF :** mean and standard deviation of false positives.

as inliers but in reality they were outliers. To do this, we first calculated the mean and standard deviation of these false positives, report in the Table 5.50.

Then, we analyzed the individual features using histogram plots. Also in this case we noticed that most of the features assumed, in correspondence with these observations, values present in both classes. The only features that could help classify some of these observations correctly were "*Src_Port*", "*Dst_Port*", "*Fwd_Header_Len*" and "*Bwd_Header_Len*". In fact, as we can see from the histograms shown in the Figure 5.11, the some inliers assume specific values only in correspondence with these false positives, while other outliers assume the same value. This could help understand that the behavior of these observations differed from that of the inliers.

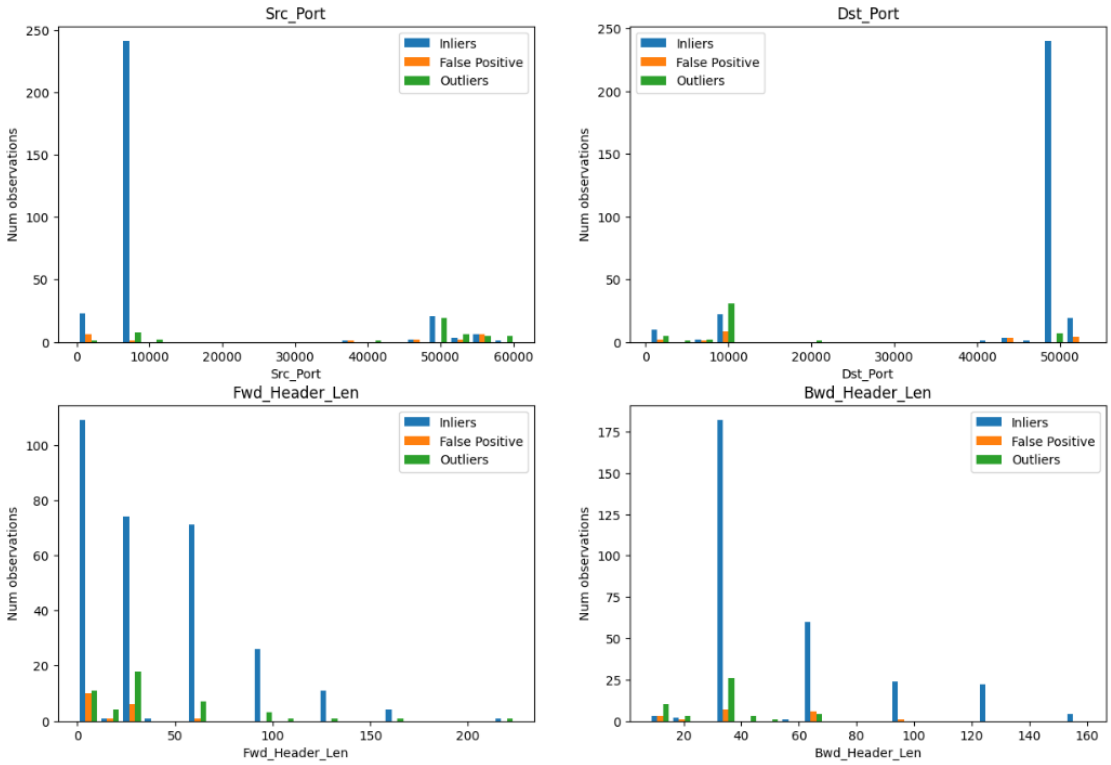

Figure 5.11. **IoT Network Intrusion dataset- Sequential RBDA-COF:** histograms of *Src_Port*, *Dst_Port*, *Fwd_Header_Len* and *Bwd_Header_Len* features (inliers in blue, false positives in orange and outliers in green).

Chapter 6

Conclusion

Anomaly detection problems are really important in many areas and increasingly studied. The anomalies are characterized by different patterns then the rest of the observations of the dataset and tendentially, the techniques of anomaly detection identify quite well a specific type of anomaly, but real datasets are often characterized by different types of anomalies. Therefore, such algorithms often fail because they are not able to correctly identify a large number of outliers. The anomaly detection algorithms described in this work, as we have seen, are classified in rank-based, distance-based and density-based. The algorithms belonging to the first group that we analyzed are RBDA and RADA, those belonging to the second group are KNN and LOCI, while those belonging to the third group are LOF and COF. To overcome the problem of the presence of different types of anomalies we considered the ensemble methods. These methods are constructed using multiple algorithms. In particular, different algorithms return an anomaly score for each observation and the results are then combined with various techniques obtaining a final decision for each observation that is taken jointly by all the considered algorithms. The ensemble methods that we have analyzed and implemented are categorized into independent and sequential. In independent methods, algorithms are supposed to be independent of each other even if this assumption is not entirely realistic. Then, each algorithm, applied to the entire dataset, returns a vector with anomaly scores of observations. Subsequently, using a combination technique, such as mean or maximum of anomaly scores and majority vote, these vectors are combined to obtain a vector containing the final anomaly score for each observation. In sequential methods there is instead a dependence between the different learning sequences and therefore between the various components. This dependence decreases with increasing randomness even if it does not disappear completely because the basic algorithms are perturbations of the same general idea. Randomness is due to the choice of random sub samples used as datasets on which algorithms are applied in the

various iterations. Sequential methods, as we have seen, can be single-layer or multi-layer. In single-layer, the output of the first algorithm, applied to the entire dataset, is the input of the next algorithm and the sequence proceeds in this way. In multi-layer methods, previous ensemble methods generate intermediate results that are then combined by another algorithm to obtain the final decision. A key characteristic of ensemble methods is that the algorithms used must be very different from each other because different algorithms will make different errors, therefore combining them, they tend to correct the errors of others. In the fifth chapter, in which the experiments carried out are reported, the two key points of this work are highlighted. In fact, for all three datasets that we used in the experiments, we demonstrated that it is possible to build ensemble methods, both sequential single-layer and independent, which return better performances than the base anomaly detection learners with which they are implemented, i.e. they more accurately identify both anomalies and inliers. The other key point we showed is that the best ensemble methods are those obtained using algorithms belonging to different classes, as we expected. Moreover, experiments have shown that the best ensemble methods are sequential. In fact, in the second layer of these methods we didn't use the entire dataset, but a sub sample composed of 30% of the observations that the first algorithm of the sequence considered more abnormal.

Therefore, using a new dataset formed by the most abnormal observations decreases the dependency between the components and helps the next algorithm to better detect anomalies, i.e. to obtain better performances. Therefore, we have shown that ensemble methods detect anomalies better than individual anomaly detection algorithms. However, we have noticed that some anomalous observations are not correctly identified. This happens because such ensemble methods fail to identify anomalies when most features assume values present in both classes or characteristic of inliers. In this case the few features with a different patterns are not enough to correctly identify the anomalies.

Appendix A

Features dataset

Features KDD CUP 99 dataset		
feature name	*description*	*type*
duration	length (number of seconds) of connection	C
protocol_type	type of the protocol	D
src_bytes	number of data bytes from source to destination	C
dst_bytes	umber of data bytes from destination to source	C
land	1 if connection is from/to the same host/port; 0 otherwise	D
wrong_fragment	number of "wrong" fragments	D
urgent	number of urgent packets	C
hot	number of "hot" indicators	C
num_failed_logins	number of failed login attempts	C
logged_in	1 if successfully logged in; 0 otherwise	D
num_compromised	number of "compromised" conditions	C
root_shell	1 if root shell is obtained; 0 otherwise	D
su_attempted	1 if "su root" command attempted; 0 otherwise	D
num_root	number of "root" access	C
num_file_creations	number of file creation operations	C
num_shells	number of shell prompts	C
num_access_files	number of operations on access control files	C
is_guest_login	1 if the login is a "guest" login; 0 otherwise	D
count	number of connections to the same host as the current connection in the past two seconds	C

Table A.1. Name, description and type of features of KDD CUP 99 dataset used in the experiments.

Features KDD CUP 99 dataset		
feature name	*description*	*type*
srv_count	number of connections to same service as current connection in past two seconds	C
serror_rate	% of connections that have "SYN" errors	C
srv_serror_rate	% of connections that have 'SYN' errors	C
rerror_rate	% of connections that have R̈EJ" errors	C
srv_rerror_rate	% of connections that have R̈EJ" errors	C
same_srv_rate	% of connections to the same service	C
diff_srv_rate	% of connections to different services	C
srv_diff_host_rate	% of connections to different hosts	C
dst_host_count	count of connections having same dst host	C
dst_host_srv_count	count of connections having same dst host and using same service	C
dst_host_same_srv_rate	% of connections having same dst port and using same service	C
dst_host_diff_srv_rate	% of different services on current host	C
dst_host_same_src_port_rate	% of connections to current host having same src port	C
dst_host_srv_diff_host_rate	% of connections to same service coming from diff. hosts	C
dst_host_serror_rate	% of connections to current host that have an S0 error	C
dst_host_srv_serror_rate	% of connections to current host and specified service that have an S0 error	C
dst_host_rerror_rate	% of connections to current host that have an RST error	C
dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error	C
label	normal or name of attack	D

Table A.2. Name, description and type of features of KDD CUP 99 dataset used in the experiments.

Deleted features of KDD CUP 99 dataset		
feature name	*description*	*type*
service	network service on the destination	D
flag	normal or error status of connection	D
num_outbound_cmds	number of outbound commands in an ftp session	C
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	D

Table A.3. Name, description and type of deleted features of KDD CUP 99 dataset.

Features Simargl2022 dataset		
feature name	*description*	*type*
FLOW_ID	Unique ID	C
L4_SRC_PORT	Source Port	C
L4_DST_PORT	Destination Port	C
FIRST_SWITCHED	Time of appearance of the first flow	C
FLOW_DURATION_MILLISECONDS	Duration of flow expressed in milliseconds	C
LAST_SWITCHED	Time of the last packet	C
PROTOCOL	Protocol flag	C
TCP_FLAGS	Number of TCP flags	C
TCP_WIN_MAX_IN	Maximum incoming TCP window size	C
TCP_WIN_MAX_OUT	Maximum outgoing TCP window size	C
TCP_WIN_MIN_IN	Minimum incoming TCP window size	C
TCP_WIN_MIN_OUT	Minimum outgoing TCP window size	C
TCP_WIN_MSS_IN	Incoming TCP segment size	C
TCP_WIN_SCALE_IN	Incoming TCP scale size	C
TCP_WIN_SCALE_OUT	Outgoing TCP scale size	C
SRC_TOS	Sets the service type byte on entry to the incoming interface	C
DST_TOS	Sets the service type byte on outgoing on the incoming interface	C
TOTAL_FLOWS_EXP	Total number of exported flows	C
MIN_IP_PKT_LEN	The smallest length of the observed packet	C
MAX_IP_PKT_LEN	The largest length of the observed packet	C
TOTAL_PKTS_EXP	Total number of exported packet	C
TOTAL_BYTES_EXP	Total number of exported bytes	C
IN_BYTES	Number of incoming bytes	C
IN_PKTS	Number of incoming packets	C
OUT_BYTES	Outgoing bytes	C
OUT_PKTS	Outgoing packets	C
ALERT	None if is inlier, PortScanning if is outlier	D

Table A.4. Name, description and type of features of Simargl2022 dataset used in the experiments.

Deleted features of Simargl2022 dataset		
feature name	*description*	*type*
PROTOCOL_MAP	Name of protocol	D
IPV4_SRC_ADDR	Source IP V4 adress	D
IPV4_DST_ADDR	Destination IP V4 address	D
ANALYSIS_TIMESTAMP	Timestamp	C

Table A.5. Name, description and type of deleted features of Simargl2022 dataset.

Features of IoT Network Intrusion dataset					
feature name	*description*	*type*	*feature name*	*description*	*type*
Src_Port	Source Port Number	C	Dst_Port	Destination Port Number	C
Flow_Duration	Duration of the flow in Microsecond	C	Tot_Fwd_Pkts	Tot packets in the fwd direction	C
Tot_Bwd_Pkts	Tot packets in the bwd direction	C	TotLen_Fwd_Pkts	Tot-size of packet in fwd direction	C
TotLen_Bwd_Pkts	Tot-size of packet in bwd direction	C	Fwd_Pkt_Len_Max	Max-size of packet in fwd direction	C
Fwd_Pkt_Len_Min	Min-size of packet in fwd direction	C	Fwd_Pkt_Len_Mean	Mean size of packet in fwd direction	C
Fwd_Pkt_Len_Std	Std-size of packet in fwd direction	C	Bwd_Pkt_Len_Max	Max-size of packet in bwd direction	C
Bwd_Pkt_Len_Min	Min-size of packet in backward direction	C	Bwd_Pkt_Len_Mean	Mean-size of packet in bwd direction	C
Bwd_Pkt_Len_Std	Std-size of packet in bwd direction	C	Flow_Pkts/s	# of flow packets per second	C
Flow_IAT_Mean	Mean-time btw. two packets sent in the flow	C	Flow_IAT_Std	Std-time btw. two packets sent in the flow	
Flow_IAT_Max	Max-time btw. two packets sent in the flow	C	Flow_IAT_Min	Min-time btw. two packets sent in the flow	C
Fwd_IAT_Tot	Tot-time btw. two packets sent in the fwd direction	C	Fwd_IAT_Mean	Mean-time btw. two packets sent in the fwd direction	C
Fwd_IAT_Std	Std-time btw. two packets sent in the fwd direction	C	Fwd_IAT_Max	Max-time btw. two packets sent in the fwd direction	C
Fwd_IAT_Min	Min-time btw. two packets sent in the fwd direction	C	Bwd_IAT_Tot	Tot-time btw. two packets sent in the bwd direction	C
Bwd_IAT_Mean	Mean-time btw. two packets sent in the bwd direction	C	Bwd_IAT_Std	Std-time btw. two packets sent in the bwd direction	C
Bwd_IAT_Max	Max-time btw. two packets sent in the bwd direction	C	Bwd_IAT_Min	Min-time btw. two packets sent in the bwd direction	C
Fwd_PSH_Flags	# of times the PSH flag was set in packets travelling in the fwd direction	D	Fwd_Header_Len	Tot bytes used for headers in the fwd direction	C
Bwd_Header_Len	Tot bytes used for headers in the bwd direction	C	Fwd_Pkts/s	# of fwd packets per second	C

Table A.6. Name, description and type of features of IoT Network Intrusion dataset used in the experiments.

Features of IoT Network Intrusion dataset					
feature name	*description*	*type*	*feature name*	*description*	*type*
Bwd_Pkts/s	# of bwd packets per second	C	Pkt_Len_Min	Min lgth of a packet	C
Pkt_Len_Max	Max lgth of a packet	C	Pkt_Len_Mean	Mean lgth of a packet	C
Pkt_Len_Std	Std lgth of a packet	C	Pkt_Len_Var	Variance lgth of a packet	C
FIN_Flag_Cnt	# of packets with FIN	D	SYN_Flag_Cnt	# of packets with SYN	D
RST_Flag_Cnt	# of packets with RST	D	PSH_Flag_Cnt	# of packets with PUSH	D
ACK_Flag_Cnt	# of packets with ACK	D	URG_Flag_Cnt	# of packets with URG	D
ECE_Flag_Cnt	# of packets with ECE	D	Down/Up_Ratio	Down/up-load ratio	C
Pkt_Size_Avg	Avg size of packet	C	Fwd_Seg_Size_Avg	Avg size observed in the fwd direction	C
Bwd_Seg_Size_Avg	Avg # of bytes bulk rate in the bwd direction	C	Subflow_Fwd_Pkts	The avg # of packets in a sub flow in the fwd direction	C
Subflow_Fwd_Byts	The avg # of bytes in a sub flow in the fwd direction	C	Subflow_Bwd_Pkts	The avg # of packets in a sub flow in the bwd direction	C
Subflow_Bwd_Byts	The avg # of bytes in a sub flow in the bwd direction	C	Init_Fwd_Win_Byts	The tot # of bytes sent in initial window in the fwd direction	C
Init_Bwd_Win_Byts	The tot # of bytes sent in initial window in the bwd direction	C	Fwd_Act_Data_Pkts	Count of packets with at least 1 byte of TCP data payload in the fwd direction	C
Fwd_Seg_Size_Min	Min segment size observed in the fwd direction	C	Active_Mean	Mean time a flow was active before becoming idle	C
Active_Std	Std time a flow was active before becoming idle	C	Active_Max	Max time a flow was active before becoming idle	C
Active_Min	Min time a flow was active before becoming idle	C	Idle_Mean	Mean time a flow was idle before becoming active	C
Idle_Std	Std time a flow was idle before becoming active	C	Idle_Max	Max time a flow was idle before becoming active	C
Idle_Min	Min time a flow was idle before becoming active	C	Label	Anomaly or Normal	D

Table A.7. Name, description and type of features of IoT Network Intrusion dataset used in the experiments.

Deleted features of IoT Network Intrusion dataset					
feature name	*description*	*type*	*feature name*	*description*	*type*
Protocol	Internet Protocol used	D	Timestamp	Timestamp of the packet	D
Flow_Byts/s	Number of flow bytes per second	C	Bwd_PSH_Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)	D
Fwd_URG_Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)	D	Bwd_URG_Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)	D
CWE_Flag_Count	Number of packets with CWE	D	Fwd_Byts/b_Avg	Average number of bytes bulk rate in the forward direction	C
Fwd_Pkts/b_Avg	Average number of packets bulk rate in the forward direction	C	Fwd_Blk_Rate_Avg	Average number of bulk rate in the forward direction	C
Bwd_Byts/b_Avg	Total size of packet in forward direction	C	Bwd_Pkts/b_Avg	Average number of bytes bulk rate in the backward direction	C
Bwd_Blk_Rate_Avg	Average number of packets bulk rate in the backward direction	C	Flow_ID	Flow Identifier	C
Src_IP	Source IP Address	C	Dst_IP	Destination IP Address	C
Sub_Cat	Sub-category of attack or Normal	D	Cat	Category of attack or Normal	D

Table A.8. Name, description and type of deleted features of IoT Network Intrusion dataset.

Bibliography

[1] Charu C Aggarwal. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, 14(2):49–58, 2013.

[2] Preeti Aggarwal and Sudhir Kumar Sharma. Analysis of kdd dataset attributes-class wise for intrusion detection. *Procedia Computer Science*, 57:842–851, 2015.

[3] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD explorations newsletter*, 2(2):81–85, 2000.

[4] Silvia Cateni, Valentina Colla, and Marco Vannucci. A fuzzy system for combining different outliers detection methods. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, AIA*, volume 2009, pages 87–93, 2009.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41, 07 2009. doi:10.1145/1541880.1541882.

[6] Alvin Chiang, Esther David, Yuh-Jye Lee, Guy Leshem, and Yi-Ren Yeh. A study on anomaly detection ensembles. *Journal of Applied Logic*, 21:1–13, 2017.

[7] Anjum Farah. *Cross Dataset Evaluation for IoT Network Intrusion Detection*. PhD thesis, The University of Wisconsin-Milwaukee, 2020.

[8] Yasir Hamid, Veeran Ranganathan Balasaraswathi, Ludovic Journaux, and Muthukumarasamy Sugumaran. Benchmark datasets for network intrusion detection: A review. *Int. J. Netw. Secur.*, 20(4):645–654, 2018.

[9] Douglas M. Hawkins. *Identification of outliers / D.M. Hawkins*. Chapman and Hall London ; New York, 1980.

[10] Huaming Huang. *Rank based anomaly detection algorithms*. PhD thesis, Syracuse University, 2013.

[11] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[12] Edwin M Knox and Raymond T Ng. Algorithms for mining distance based outliers in large datasets. In *Proceedings of the international conference on very large data bases*, pages 392–403. Citeseer, 1998.

[13] Tomáš Krutý. Ensembles for anomaly detection [online], 2018 [cit. 2022-10-05]. SUPERVISOR : Lubomír Popelínský. URL: https://is.muni.cz/th/jy4vf/.

[14] Chilukuri K. Mohan Mehrotra, Kishan G. and HuaMing Huang. *Anomaly Detection Principles and Algorithms.(1st ed.2017)*. Terrorism, Security, and Computation. Cham : Springer International Publishing : Imprint: Springer, 2017.

[15] Ibomoiye Domor Mienye and Yanxia Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149, 2022. doi:10.1109/ACCESS.2022.3207287.

[16] Maria-Elena Mihailescu, Darius Mihai, Mihai Carabas, Mikołaj Komisarek, Marek Pawlicki, Witold Hołubowicz, and Rafał Kozik. The proposition and evaluation of the roedunet-simargl2021 network intrusion detection dataset. *Sensors*, 21(13), 2021. URL: https://www.mdpi.com/1424-8220/21/13/4319, doi:10.3390/s21134319.

[17] D.S. Moore and G.P. McCabe. *Introduction to the Practice of Statistics*. Introduction to the Practice of Statistics. W.H. Freeman, 1999. URL: https://books.google.it/books?id=-_DEQgAACAAJ.

[18] Keith Noto, Carla Brodley, and Donna Slonim. Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery*, 25:109–133, 2012.

[19] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, pages 315–326. IEEE, 2003.

[20] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 427–438. ACM, 2000. doi:10.1145/342009.335437.

[21] Ralf C Staudemeyer and Christian W Omlin. Extracting salient features for network intrusion detection using machine learning methods. *South African computer journal*, 52(1):82–96, 2014.

[22] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the jam project. 09 1999.

[23] Salvatore J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144. IEEE, 2000.

[24] Pei Sun. *Outlier detection in high dimensional, spatial and sequential data sets.* Citeseer, 2006.

[25] Imtiaz Ullah and Qusay H Mahmoud. A scheme for generating a dataset for anomalous activity detection in iot networks. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33*, pages 508–520. Springer, 2020.

[26] M.Sri Vidya and G. R. Sakthidharan. Accurate anomaly detection using various machine learning methods for iot devices in indoor environment. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 308–316, 2021. doi:10.1109/I-SMAC52330.2021.9640962.

[27] Yasmen Wahba, Ehab ElSalamouny, and Ghada ElTaweel. Improving the performance of multi-class intrusion detection systems using feature reduction. *arXiv preprint arXiv:1507.06692*, 2015.

[28] Zhiruo Zhao. *Ensemble methods for anomaly detection.* PhD thesis, Syracuse University, 2017.

[29] Tommaso Zoppi, Andrea Ceccarelli, and Andrea Bondavalli. On algorithms selection for unsupervised anomaly detection. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 279–288, 2018. doi:10.1109/PRDC.2018.00050.

Acknowledgements

Before proceeding with the discussion, I would like to dedicate a few lines to all those who have supported me during my university career and thesis writing.

First of all, I would like to thank my supervisor Gianluca Mastrantonio for his valuable advice, that were fundamental for the writing of this work, and for his availability. I thank my Tutor Francesco Saracco for his precious help in conducting the research, which is the subject of my thesis, at the Data Reply company.

I am infinitely grateful to my parents, who have always motivated me to give my best. Thank you for the support you gave me, especially in times of discouragement, thank you for allowing me to undertake this path. Special thanks to my sister Chiara, my half, my greatest point of reference, and to my brother Dario, for always taking care of me. I thank Matteo, for having believed in me, always transmitting me an immense strength, helping me and enduring my mood swings. I thank all my large family for being by my side at all times. I thank my aunt Giovanna, hinge of my family, and my aunt Ninetta, an example of patience and kindness. Thanks to the little men and the princesses of my life, Tommaso, Gioele, Adele, Davide and Elsa who with your spontaneity have always snatched a smile from my lips. Thanks to my best friend Paola for supporting me and listening when I needed it. Thanks to my roommates Federica and Marta and my dear friend Sara because they have been my second family in these years away from home. Thanks to my colleagues, especially Gioana, with whom I spent days and nights studying, but also leisure moments. Finally, a heartfelt thanks to my grandparents, who today cannot be here with me, but I am sure they are watching me from up there.