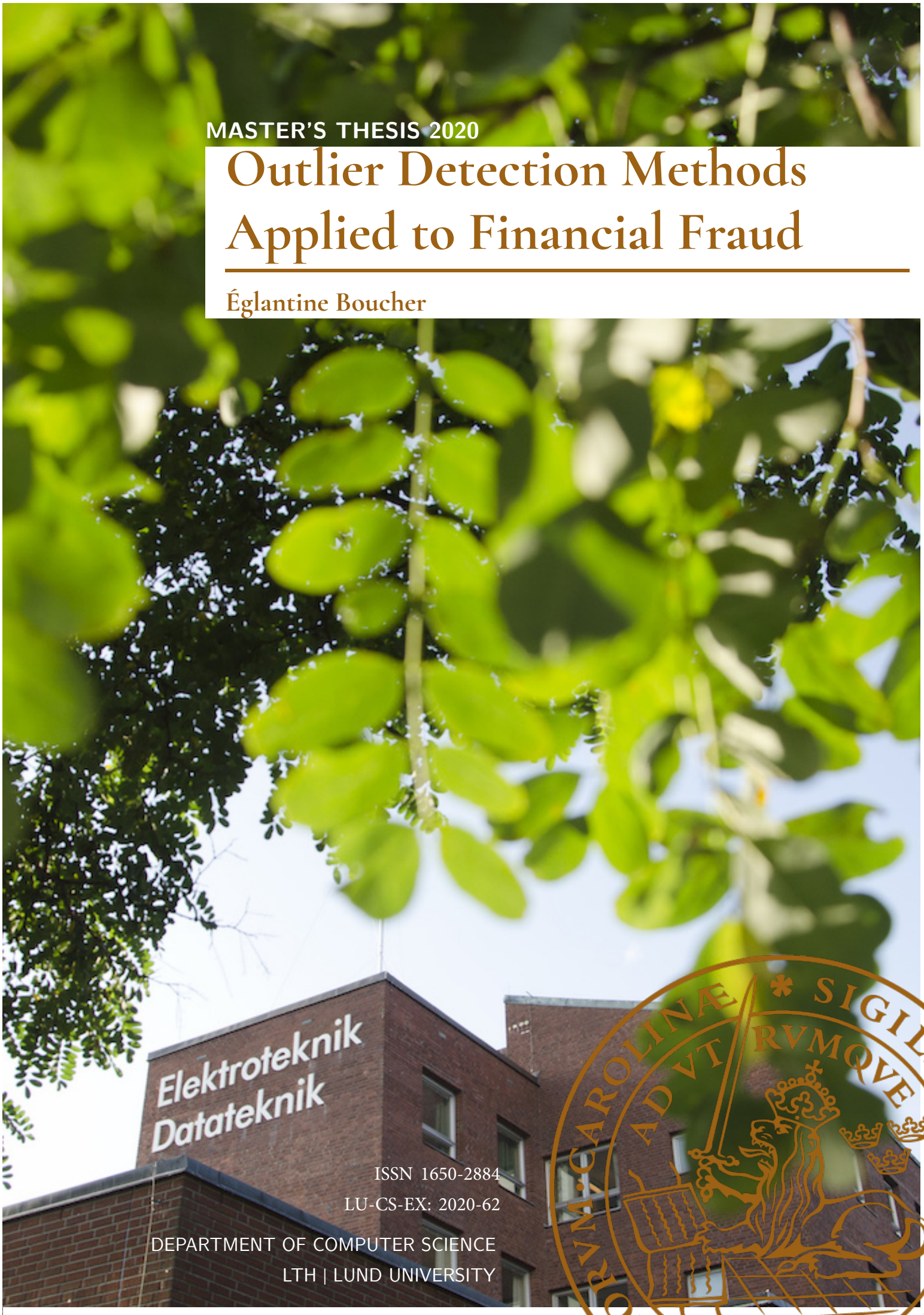
MASTER'S THESIS 2020

Outlier Detection Methods Applied to Financial Fraud

Églantine Boucher



ISSN 1650-2884
LU-CS-EX: 2020-62

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-62

Outlier Detection Methods Applied to Financial Fraud

Églantine Boucher

Outlier Detection Methods Applied to Financial Fraud

Églantine Boucher
eg2687bo-s@student.lu.se

August 20, 2020

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Pierre Nugues, pierre.nugues@cs.lth.se

Examiner: Jörn Janneck, jorn.janneck@cs.lth.se

Abstract

Nowadays, most transactions are made in a dematerialized way. Therefore, it becomes more and more important to find efficient ways to detect fraudulent transactions. Currently, this is done by conventional, rule-based techniques even if big structures are transitioning to machine learning techniques.

The aim of this thesis is to see if outlier detection algorithms are an efficient way to detect fraud. To do so, I have tested two naive and four machine learning methods on three sets of different size and features. They represent a financial banking set, a credit card set, and a company audit set.

With logistic regression, I achieved an improvement for the F-1 score of 32% with the credit card set and 45% with the audit set. Moreover, the results with the audit set are good for all methods. As a conclusion, logistic regression is a relevant method for fraud detection especially for a small balanced set (less than 1000 rows) even if with the other data sets the results are also acceptable.

Keywords: Fraud, machine learning, finance, credit card, audit, logistic regression

Acknowledgements

I would like to thank everyone that has contributed to this thesis and made this work possible.

I would like to thank my supervisor Pierre Nugues for his dedication week after week to my work and for his considerably helpful advice. Thank you also to my supervisor Jörn Janneck for trusting my work.

A big thank you to Miloud Belkacem and Ada Voka who have given me the technical support and context I needed.

I would like to give a special thank to my family and friends for having supported me during my two years in Sweden. And, I am incredibly grateful to my home university, École Centrale de Marseille, for having made all of this possible.

Contents

Chapter 1
Introduction

1.1 Background and motivation

In the current banking system, all around the world, most transactions are made via credit card or bank transfers. We can see all these electronic transactions as raw data that could be sorted and analyzed in order to extract information for the bank. It is often said that a key to efficient business is customer insight and this implies the knowledge of transactions (Laughlin, 2014).

This data analysis of financial transactions can be further used in a commercial goal: In order to expand the bank profit or to detect fraudulent transactions made from one bank account to another. For the latter, many tools are available to detect such transactions in the billions of regular ones, and notably outliers detection techniques. By definition, an outlier is a value in a data set that differs significantly from the others and shows a contrast. A fraudulent transaction can then be detected by its contrast with other transactions, for instance by the transaction time or its amount.

In this thesis, I evaluate *outliers detection techniques* on three different data sets to determine the properties and effectiveness of these methods in different conditions. The datasets are:

1. A large set of six million rows with less than 1% of fraud,
2. A medium set of 200,000 rows with less than 1% fraud too, and
3. A small balanced set of 800 rows.

Having three sets like these ones with a different size, number of features, and percentage of fraud should reflect, in my opinion, a good data diversity to test the algorithms.

The aim of this thesis can be rephrased as the following questions:

1. Are outliers detection techniques accurate to detect fraudulent banking transactions?

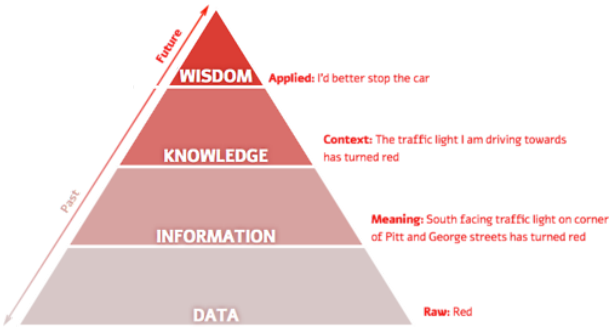


Figure 1.1: The DIKW pyramid

2. Which improvement do we get by using machine-learning techniques instead of rule-based ones?

1.2 About data

According to Bernstein (2009), data, information, knowledge, and wisdom form a pyramidal scheme. Data is an observation, which has no value before being processed to become an information. It then becomes an instruction and gives knowledge. This finally leads to wisdom and the ability to see and predict long term consequences. Therefore, raw data such as banking transactions can become wisdom through the entire process of filtration, reduction, and refinement, which are different steps of *data management*.

Indeed, data has no interest in being collected infinitely without any transformation. It is just the first step in the entire process of data analysis, whose final goal is to predict the future and improve our knowledge of the data. This process is called *data mining*.

Data mining is a process aiming at discovering patterns in a data set using different techniques of mathematics and computer science. The idea is to find information out of data to discover knowledge from the database. According to Han et al. (2011), the different steps of the entire data mining process are the following:

1. **Data cleaning:** Clean the data base by removing inconsistent data.
2. **Data integration:** Gather data from different sources.
3. **Data selection:** Select data relevant for the topic.
4. **Data transformation:** Transform data to process with mining.
5. **Data mining:** Extract models by using mathematical and computational techniques.
6. **Pattern evaluation:** Evaluate models useful for the knowledge.
7. **Knowledge presentation:** Present knowledge acquired to the users.

Furthermore, with all data banking facilities collect rule based techniques to detect fraud seem outdated and should be replaced by machine learning methods. Sithic and Balasubramanian wrote a survey of insurance fraud using data mining techniques. They concluded that outlier detection and clustering algorithms appear as efficient methods.

Therefore, in this thesis, I will focus my work on the fifth step of the data mining process and I will use outlier detection techniques applied to financial data sets.

1.3 Financial fraud

Currently, in the payment system, two different types of fraud detection models are used:

- Historically, the first detection techniques were based on rules and identification schemes that have been set manually. These methods are moderately accurate and take a lot of time;

- Machine learning-based fraud detection which aims at detecting fraud automatically in or near real time by identifying hidden correlation in the data.

In practice, conventional rule-based fraud detection techniques are still prevalent as most firms still use legacy systems. Nonetheless, bigger companies are starting to move on to machine learning techniques (SAS, 2020).

Most fraudulent transactions happen with credit cards all around the world. There are two types of frauds:

1. Card present frauds; for instance counterfeit card, lost and stolen card, and identity takeover; and

2. card not present fraud; for instance when hackers manage to get banking information in a non face to face setting.

In this thesis, I will focus my work on the second type of fraud *i.e.* fraud that occurs on the internet, for instance e-commerce.

According to Chuprina (2020), a good mean to prevent these fraudulent transactions is data mining and pattern recognition, for example using clustering methods. For this, the more transaction samples we have, the most effective these methods are. Indeed, there are a few requirements for AI-based fraud detection methods. The first one is the amount of data: models trained on a large mount of data generally have a better accuracy. The second one is the quality of data, and especially data sorted correctly.

Furthermore, there are two main types of machine learning techniques: unsupervised and supervised. The first category does not need any labeling to find patterns in the data in contrast to the second one. In this thesis, I will use supervised algorithms: local outlier factor, and support vector machines; and unsupervised algorithm: isolation forest. I will also implement two naive algorithms: clustering and statistical method because they are currently the most used models in financial fraud detection (Chuprina, 2020).

Even if I chose to work with labeled sets, I will also apply unsupervised algorithms in this work. A reason that justifies the relevance of unsupervised methods is because real life sets are not labeled and the aim is to have a result as close to real life as possible.

1.4 Thesis outline

In Chapter 2, I present previous works done on outlier detection methods. First, the naive techniques: clustering and statistical methods. Then, the machine learning methods: isolation forest, local outlier factor, support vector machine, and logistic regression. Eventually, one deep learning method: neural networks.

In Chapter 3, I describe the models I implemented and how I measured their accuracy. Then, I present the three different data sets I have worked with. And finally, I show the results.

Then, in Chapter 4, I discuss previous results obtained with the data sets I have worked with and I compare my results to theirs. I then evaluate my results and outline ideas of improvement for financial fraud detection.

Eventually, Chapter 5 ends this thesis with a conclusion.

Chapter 2
Previous work

In this section, I describe several outlier detection methods. First, the clustering and statistical methods, which are two intuitive methods and then, three machine learning methods *support vector machines*, *isolation forests*, and *local outlier factors*. Additionally, I will also present a deep learning method neural networks with *multi layer perceptron classifier*.

2.1 Naive methods

Naive algorithm opposed to machine learning algorithms are intuitive algorithms coded naturally. In this section, I will introduce two types of algorithms: clustering algorithms with DBSCAN and HDBSCAN, and statistical algorithms.

2.1.1 Clustering methods

Data clustering is a particular kind of data mining problem. The aim of data clustering is to identify patterns in the entire distribution of the data set. In some one dimensional or two dimensional cases, a plot can be made and the patterns appear quite clearly.

Let us take the following example to illustrate the idea behind the clustering method. Figure 2.1 shows a set of random points, where we apply the K-means algorithm. K-means will group the points according to their distance to each other. The algorithm has one hyperparameter that must be set before we run it: the number of clusters. In the left part of Fig. 2.1, we apply the k-means with three clusters, and in the right part with six.

According to Zhang et al. (1996), there exists multiple clustering methods using different approaches. Indeed, to link a point to a cluster, we can use two methods:

1. A probability-based approach;

2. A distance-based one.

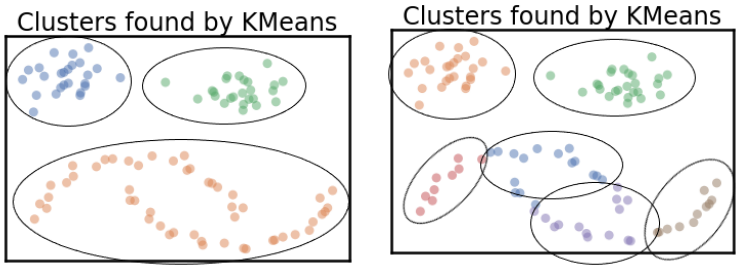


Figure 2.1: The same random data set split in three and in six clusters

In this section, I will describe an outlier detection method based on statistics so I decided to use a distance-based approach for the pure clustering method.

Most popular clustering algorithms can be split in two subcategories regarding our current problem that is outlier detection:

1. The first category requires the number of clusters to be given in parameters. this include the K-means, affinity propagation, spectral clustering and agglomerative clustering algorithms. For example, Figure 2.1 shows the same data sets split in a different number of clusters;

2. In the other one, we have DBSCAN, and HDBSCAN algorithms.

The point of this thesis is to compare multiple outlier detection methods. In this context, we can argue that the corresponding outliers in the data set should not be assigned to a cluster. It is therefore not possible to take the number of clusters as a parameter. That is why only the algorithms of the second subcategory will be relevant. Therefore, I will investigate the DBSCAN and HDBSCAN method.

DBSCAN. The *density-based spatial clustering application with noise* (DBSCAN) was first presented at the KDD'96 data mining conference (Ester et al., 1996). This algorithm has directly become very popular and is now used among many real-world applications. For an additional review of this algorithm, see Schubert et al. (2017).

This density based model has two key hyperparameters:

- *epsilon:* The maximum distance between two points to consider them as neighbors;

- *min points:* Minimum number of points to form a cluster.

The choice of these parameters plays a large role in the accuracy of the model. Indeed, if the *epsilon* is too large, all the points might end up in a cluster and the outlier detection will be complicated. Just as if *min points* is too large, we might end up without any cluster at all and a data set only formed by anomalies.

For experimental evaluations, Ester et al. (1996) recommend to take *min points* equals to twice the number of dimension of the data set. For instance, as in the example below (Figure 2.2) if the data set is 2-dimensional *min points* will preferably be four. The determination of *epsilon* is less straightforward and depends on the percentage of noise (*i.e.* outlier points) wanted. This variable will then be derived depending on the experimental data set we are working on.

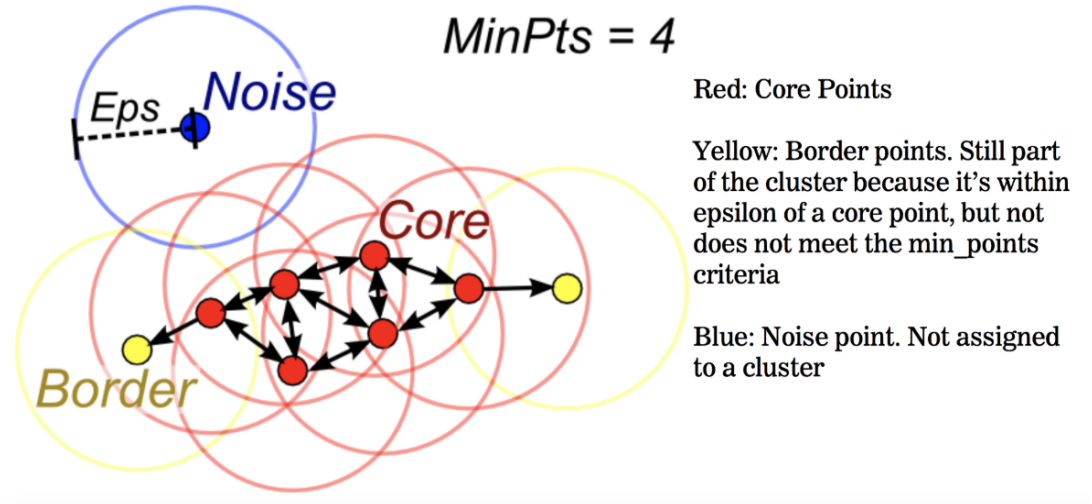


Figure 2.2: Outline of the DBSCAN algorithm

To understand this model better, let us take Fig. 2.2 from Lutins (2017).

This figure is quite explicit: Dots represent points in the data set and the circle around them represents the given *epsilon* distance. Here the minimum number of points needed to form a cluster is four. Therefore, when screening the data set, if DBSCAN finds four points within *epsilon* distance of each other, it is a cluster.

To go a bit further in the explanation, Schubert et al. (2017) gives an abstract algorithm and a pseudocode of the DBSCAN algorithm.

The abstract DBSCAN algorithm can be enumerated as follows:

1. Compute neighbors of each point and identify core points;
2. Join neighboring core points into clusters;
3. for each non-core point:
   (a) Add to a neighboring core point if possible;
   (b) Otherwise, add to noise.

More precisely, Schubert et al. (2017) provides a simplified pseudo code of this algorithm shown in Figure 2.3.

The DBSCAN algorithm has multiple advantages compared to other clustering algorithms. As said before, it doesn't require the number of clusters to be given as a parameter. This makes it a very useful algorithm for outlier detection. As it is a density based algorithm, it is great to use when having clusters of different densities within the same data set. It can also find arbitrary shaped clusters, which is very difficult for other clustering algorithms.

The choice of *min points* and *epsilon* can either be an advantage or a drawback. If the data set is well understood, then it is really useful to be able to choose these parameters. However, if the set is not well understood or the differences in the cluster densities too large, setting an appropriate *epsilon* and *min points* values will be complicated. And, consequently, this will impact the results of the experimentation.

ALGORITHM 1: Pseudocode of Original Sequential DBSCAN Algorithm

```
Input: DB: Database
Input: ε: Radius
Input: minPts: Density threshold
Input: dist: Distance function
Data: label: Point labels, initially undefined
foreach point p in database DB do                    // Iterate over every point
    if label(p) ≠ undefined then continue            // Skip processed points
    Neighbors N ← RangeQuery(DB, dist, p, ε)         // Find initial neighbors
    if |N| < minPts then                             // Non-core points are noise
        label(p) ← Noise
        continue
    c ← next cluster label                           // Start a new cluster
    label(p) ← c
    Seed set S ← N \ {p}                             // Expand neighborhood
    foreach q in S do
        if label(q) = Noise then label(q) ← c
        if label(q) ≠ undefined then continue
        Neighbors N ← RangeQuery(DB, dist, q, ε)
        label(q) ← c
        if |N| < minPts then continue                // Core-point check
        S ← S ∪ N
```

Figure 2.3: Simplified pseudo-code of the DBSCAN algorithm

HDBSCAN. The *Hierarchical density based spatial-clustering of application with noise* (HDBSCAN) algorithm is similar to the DBSCAN algorihtm as it is also an intuitive clustering method. Campello et al. introduced it in 2013 at the Pacific-Asia conference on knowledge discovery and data mining. The difference between the two algorithms lies in the input parameters: in the HDBSCAN algorithm only min points is taken as input.

With *min point*, this algorithm then computes a clustering tree containing all partitions obtainable by DBSCAN for different values of *epsilon* in a hierarchical way.

It also contains nodes that indicate when a point changes from being a core point to a noise point (i.e. an outlier). Indeed, knowing the value of the minimum of points required to make a cluster, as we go down in the hierarchy we can decide at each split if two clusters are formed or not. If one of the new clusters created by the split has fewer points than the minimum points required in input, these points are categorized as noise.

Thanks to this step, the clustering tree is condensed. The next step of the HDBSCAN algorithm is to extract the clusters. For this, we introduce *lambda*, which is $\lambda = \frac{1}{\epsilon}$. Each cluster is assigned a λ_{birth} the value corresponding to when the cluster splits to be its own cluster and a λ_{death}, the value corresponding to when this cluster splits into smaller clusters. We also define, for each point of a cluster, λ_p the value at which, if it happens, this point drops out of the cluster.

Then, for each cluster, a stability value can be derived as $\sum_{p \in cluster}(\lambda_p - \lambda_{birth})$. Going down the clustering tree again, we compute the stability of all clusters. If the sum of each stability of the child clusters is greater than the stability of the parent cluster, then the cluster stability is set to be the sum of its children. Otherwise, the parent cluster is selected and its children are automatically abandoned. When the root node is reached, HBDSCAN returns the set of clusters obtained.

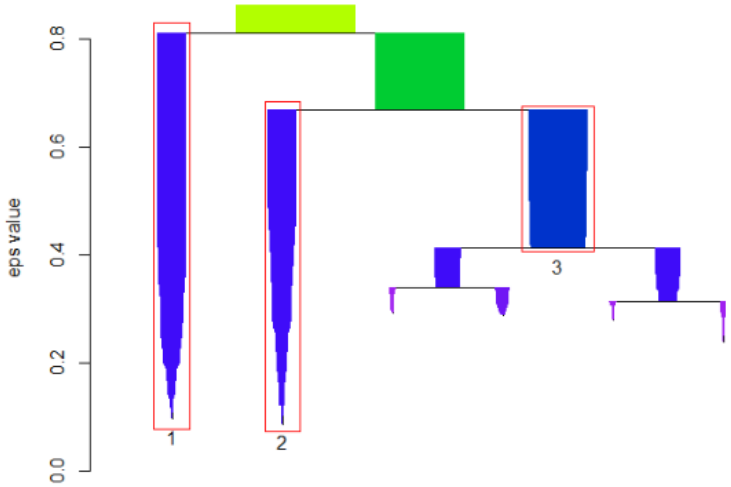


Figure 2.4: Condensed tree plot of a random set

To understand better this concept, an example is given on Figure 2.4. On this figure, the condensed tree of a random set is plot. We can see framed in red the three clusters HDBSCAN found. Indeed, for the third one as the stability of the parent cluster was bigger than the sum of the ones of its children the parent cluster was selected. It is the opposite way for the first and second cluster.

To sum up, the two clustering methods, DBSCAN and HDBSCAN, are pretty similar. The only difference lies in the choice of *epsilon* which is left to the user for DBSCAN and computed within the HDBSCAN algorithm. Depending on the ability of the user to understand the data set, one method can be better. If there is a perfect understanding of the data, DBSCAN will be preferred as we can choose *epsilon*. Otherwise, HDBSCAN might be a better option.

In this thesis, I will start the analysis of my data sets with the HDBSCAN algorithm and see how it behaves. If the results are not accurate enough, I will try if possible to switch to a different choice of *epsilon* and therefore use DBSCAN.

2.1.2 Statistical method

A first idea when thinking about a statistical method to detect outliers is the mean absolute deviation.

We can apply this technique to data sets where we assume a normal distribution. In this case, 68% of the data values are within one standard deviation of the mean, 95% are within two standards deviations, and 99.7% are within three standard deviations. Therefore, if a point of the data set is more than three standard deviation away from the mean, it is very likely to be an outlier.

The standard deviation formula is:

$$s_N = \sqrt{\frac{1}{N}\sum (x_i - x)^2}, \tag{2.1}$$

where all the x_i are the values of the data set, x its mean, and N, its size.

However, according to Leys et al. (2013) the median deviation is a better alternative than the mean absolute deviation. Indeed, in their article they compare the two techniques and show that the median absolute deviation is a solution to most problems encountered by the mean absolute deviation.

The median absolute deviation is defined as:

$$MAD = b \cdot M_i(|x_i - M_j(x_j)|), \tag{2.2}$$

where:

- x_j are the original observations;
- M_i is the median of the series;
- b is a constant linked to the distribution of the data.
  - In a normal distribution $b = 1.4826$;
  - in others distribution, $b = \frac{1}{Q(0.75}$ where $Q(0.75)$ is the 0.75 quantile of this distribution.

Then, we have to choose which values of our set are the most relevant to apply these calculations on. Indeed, it is more accurate in a multivariate data set to focus on important features instead of doing this statistical method on all features.

According to Miller, there are three suitable values for the number of MAD we want:

- Very conservative, 3;
- Moderately conservative, 2.5;
- Poorly conservative 2;

This value has to be chosen depending on our data set and its values.

2.2 Machine learning methods

A very simple way to see the difference between naive algorithms and machine learning algorithm is this:

- *naive algorithms* take rules and data as inputs and gives answers as outputs.
- *machine learning algorithm* take answers and data as inputs and give rules as outputs.

Machine learning is a part of artificial intelligence. It is nowadays used in numerous fields such as medicine, retail, or finance. The aim of machine learning is to compute the solutions to a program that human beings can't explain and therefore can't write. Alpaydin (2020) gives the example of face recognition. It is a task we do everyday without even thinking about it and if we would have to explain how we do it we couldn't. Then, it is impossible to write a computer program able to recognize face samples of a person directly. However, with machine learning, the program can capture specific pattern of a person's face such as the mouth, the eyes, and learn to recognize one from another.

What machine learning algorithms have in common is their ways of working. There is always a training set and a test set. As explained before, the aim of machine learning algorithm is to "learn" from a sample of the data set usually called training set and then, to give the output on a test set.

In this section, I will introduce three machine learning algorithms used for outlier detection: *support vector machines*, *isolation forests*, and *local outlier factors*.

2.2.1 Isolation forest

Isolation forest is especially designed for outlier detection. It was first proposed in 2008 by Liu et al..

In this algorithm, the training of the data consists of creating trees. Let's first introduce a data set of dimension N. The first step of the algorithm is to sample the training data and to construct a binary tree. The branching process of this tree works as follows:

1. Select one dimension d between 1 and N,
2. Select a random value v_r between the minimum and maximum values of this specific dimension,
3. Screening the data, if the point has a a smaller value than v_r for dimension d, the point is sent to one branch; if not, it is sent to the other branch.

With this process, each node leads to two branches. Then, the process is repeated over the entire data set until one point is isolated or a depth limit is reached.

Then, the process starts again with another sample of the initial data set. When a large number of trees is constructed, the training is complete. This algorithm works on the principle that anomalies in a data set are "few and different". Therefore, the depth of each point is a good indicator to separate outliers from regular points. Indeed, on average, outliers will have a smaller depth than regular points.

Sun et al. (2016) have a clear representation of a random partitioning of a data set and its tree, see Figure 2.5.

By creating all these trees, anomaly scores can be computed so that any new point can go through these trees with respect to the trained data. The anomaly score equation is given by:

$$s(x,n) = 2^{(}\frac{-E(h(x))}{c(n)}) \tag{2.3}$$

Where $E(h(x))$ is the mean value of the depths one data point reaches in all trees; $c(n)$ is the average depth in an unsuccessful search in a Binary Search Tree given by the following equation:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2.4}$$

Where $H(i)$ is the harmonic number approximated by $\log(i)+\gamma$, where γ is Euler's constant; and n the number of points involved in the construction of trees.

This anomaly scores are derived for each tree and averaged across the different trees to get a final anomaly score among the entire forest for a given data point. Then, if this score is close to one, this point is labeled as an outlier. If the score is closed to zero, it is labeled as a regular point.

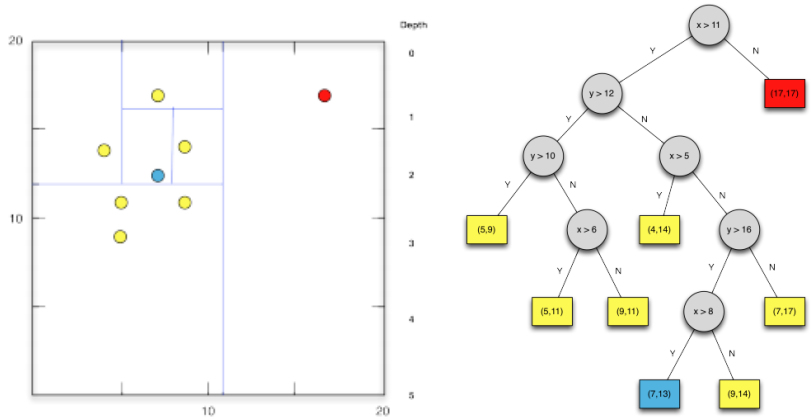


Figure 2.5: Left: a random data set, in red the anomaly. Right: the tree associated to this partitioning

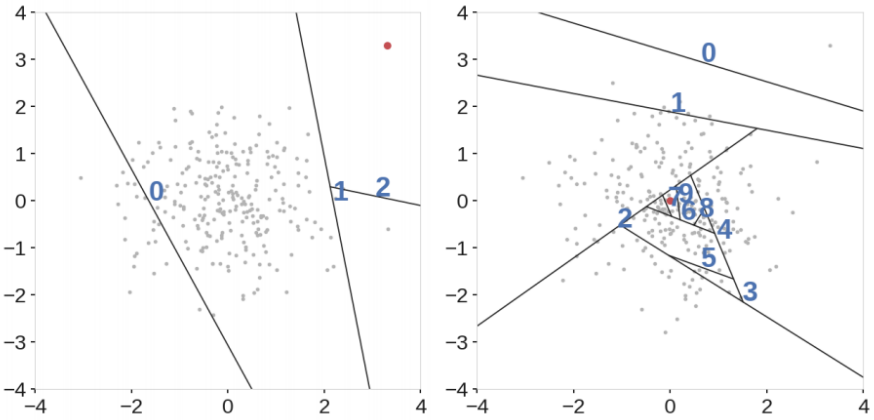


Figure 2.6: Left: an anomaly. Right: a nominal point

The isolation forest algorithm is a good method to find anomalies in a data set even if the set is small. However, the way of partitioning the set when taking branch cuts is always horizontal or vertical. This can introduce bias in the detection of anomalies. This is why in 2018, Hariri et al. introduced the *Extended Isolation Forest* algorithm. This algorithm is very similar to Isolation forest but takes random slopes as branch cut, instead of selecting a random dimension and a random value within this dimension.

Figure 2.6 shows the different partitioning with extended isolation forests on a random data set; Hariri et al. (2018).

2.2.2 Local outlier factor

The *local outlier factor* (LOF) algorithm was first introduced by Breunig et al. during the international conference on management of data in 2000. It is an algorithm specifically made for anomaly detection based on others like DBSCAN or K-nearest neighbors.

The LOF algorithm takes, as an input, an integer that we will call k. This parameter represents the number of neighbors the algorithm will consider. Indeed, the first step of

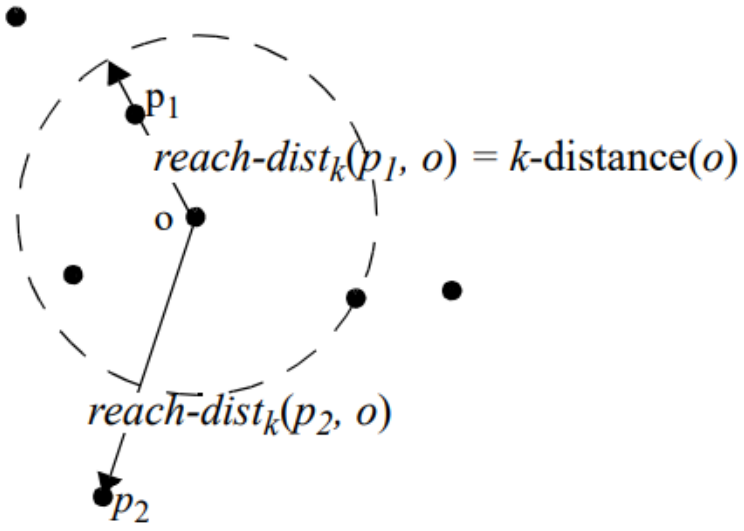


Figure 2.7: Example of rechability distances: p_1 and p_2

LOF is to calculate the density of the neighbors of a given point to compare it afterward. This parameter is important as a small k will induce a more local focus, whereas a big one can lead in missing outliers.

The k parameter is then used to calculate the k-distance which is the distance of one point to its k-th neighbor *i.e.* the k-th closest point to the given point.

With this k-distance, the reachability distance can be computed and is given by the following formula:

$$reach\_dist(a, b) = \max(k - distance(b), dist(a, b)), \tag{2.5}$$

where a and b are two points of the data set. If point a is within the k neighbors of point b, the reachability distance will be the k distance of b. Otherwise, it will be the Euclidean distance between a and b. To understand this concept better, Figure 2.7 shows the two different cases (Breunig et al., 2000).

With the reachability distance, the local reachability density (lrd) is introduced. It is derived from the formula:

$$lrd(a) = \frac{1}{\frac{\sum_n reach\_dist(a, n)}{k}}, \tag{2.6}$$

where $\sum_n(reach\_dist(a, n))$ is the sum of all the reachability distances between a and its k neighbors. Dividing by k makes it an average and the density is the inverse of that average. *lrd* is an indicator of how far a point is from its neighbors.

With this *lrd*, the LOF can be computed according to this formula:

$$LOF(p) = \frac{\sum_n \frac{lrd(n)}{lrd(p)}}{k} \tag{2.7}$$

To sum up, the LOF algorithm derives the LOF of each point in the data set. The LOF being the density of one point compared to the ones of its neighbors. If the density is smaller

than the ones of its neighbors, ($LOF >> 1$), then the point is an outlier. Otherwise, LOF should be close to or less than one.

The strength of this algorithm is its local specificity. Indeed a point close to a very dense cluster might be labeled as an outlier even if it has the same density value as points in a sparser cluster. And therefore, the LOF algorithm can detect outliers where other algorithms can't.

However, the fact that it is based on a quotient might be a drawback. Indeed, if the LOF value for a point in a very clean data set is 1.1, it might be an outlier, whereas in a more various data set, this point can be considered as an inliner. It is therefore very important to know how to interpret LOF values.

2.2.3 Support vector machine

The support vector machine (SVM) was first introduced by Vapnik in 1979, then published in 2006, and is now a popular machine learning algorithm. According to Srivastava and Bhambhu (2009), its performance is generally higher regarding classification accuracy than the other classification algorithms. SVM can be linear or non linear.

SVM is used for classification and regression. Given a training set of data, which is commonly a sample of the data set studied, this algorithm will split the points in two classes. In the case of a linear support vector machine, it constructs an optimal hyperplane if the data set in in two dimension or a set of optimal hyperplanes if the set has more dimensions.

Let's introduce a data set: $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)$, where the coordinates are:

- $\mathbf{x}$ a p dimensional vector;
- y which takes only 1 and -1 as values indicating the class of $\mathbf{x}$.

The aim of SVM is to find an optimal hyperplane to split this set into two classes. What here is called an optimal hyperplane is the linear function with maximal margin between the vectors of the two classes; see Figure 2.8 (Cortes and Vapnik, 1995).

In the linear case, the equation of the hyperplane can be written as $\mathbf{w} \cdot \mathbf{x} + b = 0$ where b is a scalar and $\mathbf{w}$, the weight, a p-dimensional vector normal to the hyperplane; p being the dimension of each point of the data.

On the other hand, the maximal margin is given by two equations: $\mathbf{w} \cdot \mathbf{x} + b = 1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$. Any point above the first hyperplane is from one class labeled 1 and any point below the second hyperplane is from the second class with label -1. The distance between these two hyperplanes being $\frac{2}{||\mathbf{w}||}$. To maximize it, we have to minimize $||\mathbf{w}||$.

We also need to prevent data points to be in this margin, therefore two others equations follows:

- $\mathbf{w} \cdot \mathbf{x_i} - b >= 1$ if $y_i = 1$
- $\mathbf{w} \cdot \mathbf{x_i} - b <= 1$ if $y_i = -1$

which can be rewritten in one equation: $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) >= 1$ for all i. What the SVM algorithm is technically doing to find the optimal hyperplane is to solve the following optimization problem:

Minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) >= 1$ for all i.

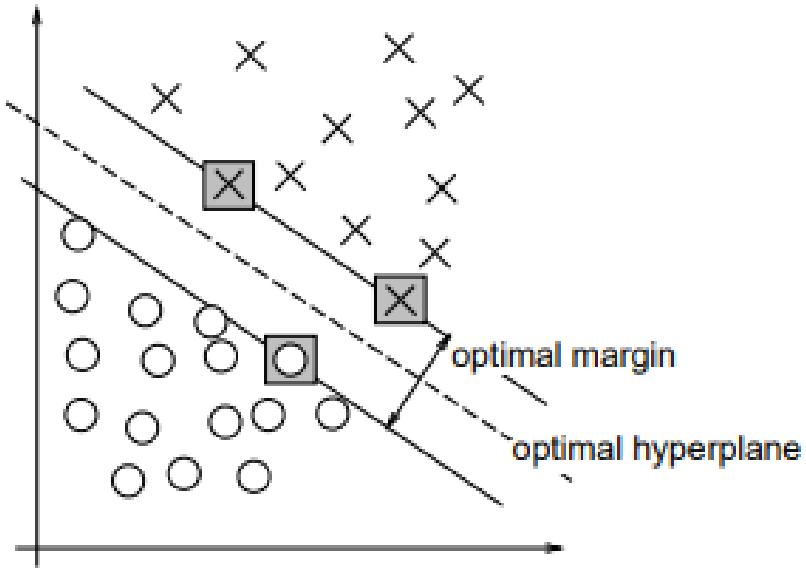


Figure 2.8: Illustration of the hyperplane separating two groups in the linear SVM algorithm

However, it occurs that in the dimension given by the initial data set, it is impossible to find the optimal hyperplane. Then, the solution is to switch to the nonlinear SVM algorithm. This algorithm is the same as the linear one except that it uses kernel functions instead of dot product in all the functions above.

2.2.4 Logistic regression

In some cases, the complexity of the SVM algorithm is too high, especially when dealing with large data sets or data with identified independent variables. Logistic regression (LR) is then preferred to SVM. Indeed, its complexity is lower and it has a mathematical model which makes it relevant.

Logistic regression was originally used in cases where the response variable Y is binary *i.e.* in cases where the outcome for the variable has only two possible types. In outlier detection for instance, the response is 1 if it is an anomaly and 0 if it is not. Then, the variance can be written $V[Y] = p(\mathbf{x})(1 - p(\mathbf{x}))$ as a function of $p(\mathbf{x}) = P(Y = 1|\mathbf{x}) = E[X]$. See Hosmer et al. (1989) for a description.

Then, we can introduce the logistic function:

$$p(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}), \tag{2.8}$$

where:

$$Logistic(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

The aim of logistic regression is to fit $\mathbf{w}$ to predict the outcome variable Y given $\mathbf{x}$. The regression curve produces the probability of the outcome to occur.

It is clearly shown on Figure 2.9 (Yiu, 2019). Indeed, the figure on the left represents all the $\mathbf{x}$ points. And, the figure on the right represents the same points split in two categories:

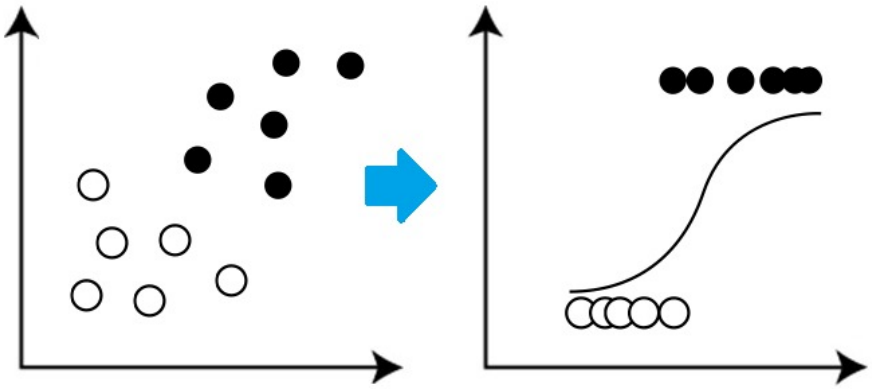


Figure 2.9: Separation of a data set with logistic regression

- the ones which probabilities are above 0.5, the threshold value;
- the one which probabilities are under.

The blue arrow represent the scalar product of Equation 2.8.

2.3 Neural networks

After naive and machine learning methods come neural networks. Neural networks are a part of deep learning which is one step ahead of machine learning techniques, the main focus of this thesis. However, I decided to add one deep learning algorithm to my work in order to have a larger comparison spectrum. The algorithm I chose is multi layer perceptron classifier (MLP).

Functional neural networks with many layers were first introduced in 1967 by Ivakhnenko and Lapa.

An artificial neural network works similarly to a biological neuron. Indeed, artificial neurons has inputs and produce outputs linked to several other neurons. And, in a multi layer neural network inputs of some neurons can be outputs of others.

As we can see on an example illustrated by Figure 2.10 a network consists of connections between all neurons of all layers (Yiu, 2019). To each connection is assigned a weight $w_{i,j}$ between input (or neuron) i and neuron (or output) j.

For the MLP in this thesis I decided to choose logistic regression. Therefore, the activation function will be the one developed in the previous section. Logistic regression equation implemented via a neural network composed of only one feature can be written as followed:

$$p(\mathbf{x_1}) = Logistic(\mathbf{w_1} \cdot \mathbf{x} + b) \tag{2.10}$$

Where $\mathbf{x_1}$ is the input, $\mathbf{w_1}$ is the weight of the connection, and b the bias.

Then, when applied to a bigger and more complex network such as the one on Figure 2.10. If we suppose the prior layer to be m elements deep and the current layer to be n elements deep, the output Y can be written as:

$$\mathbf{Y} = \mathbf{W} \cdot \mathbf{X} + \mathbf{B} \tag{2.11}$$

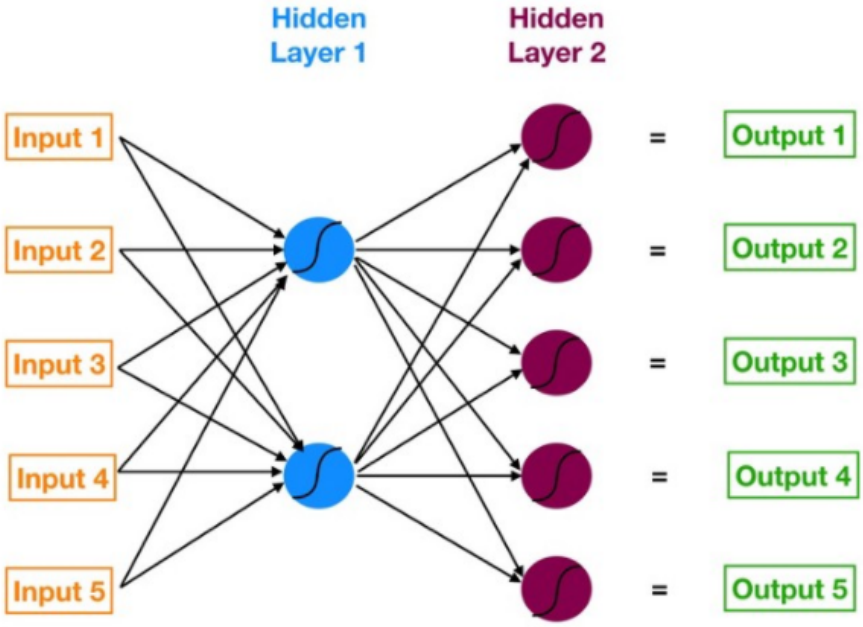


Figure 2.10: Scheme of a neural network of two hidden layers with five inputs and outputs

Where **W** is the n by m matrix of weights, **X** is the m by 1 matrix of starting inputs or outputs of the prior layer, and **B** the n by 1 matric of bias. Then, we can apply the logistic function to each element of **Y** which gives the neurons outputs for the current layer.

Then, we just repeat this operation for every layer in the network and it will finally gives the final outputs. Here, the activation function used is the one of logistic regression but if we wish to use another mathematical model it would be the same operation with another function.

Chapter 3
Methods and results

Anomaly detection algorithms have several real-life applications. Their most intuitive use is to find anomalous values and detect fraud, but it can also be used to optimize a system. For instance in public transports, detecting outliers in traffic flow distribution data allows to detect a problem on the lines and can help optimize the service.

In this thesis, I will focus on banking, financial, and audit fraud detection, and use several data sets found on open source to do my experiments. I wrote all the model implementations on Jupyter notebooks with the Python language.

In this section I will present the methods and sets I used. I decided to work on three different data sets to embrace all financial fraud. The first set is a very large set, which represents transactions in a facility like a bank. The second one is a smaller set which represents personal transactions everyone can have throughout a year for example. And, the last one represents an audit among firms and is smaller than the others.

I chose these sets because they complement each other quite well. Furthermore, their sizes are different and it is interesting to see if the algorithms I chose are sensitive to it.

3.1 Methods

With every data set, a preprocessing work is necessary before getting started with the study in itself. The aim is to check that there are no invalid values in the data set and that all data are in the correct type to run the algorithms.

In the financial data set, no invalid values were found therefore, no values had to be changed. However, not all columns are important to perform the algorithms. Indeed, the column *IsFlaggedFraud* is of no use for our models. This label indicates only the fact that the transaction is above a certain amount and has no information on the fraudulent aspect of it. Then, for the application of the algorithms the column *IsFlaggedFraud* is removed from the data set.

Besides, only numbers are to be used for our models that will not work if there are strings

as inputs. Therefore, the columns *nameOrig*, and *nameDest* are also removed from the data set. However, as we know that there are five different types of transactions I decided to do a one-hot encoding for the column *type*. Then, we can take the type into account because it can be relevant in finding outliers. But, I decided not to do this for *nameOrig*, and *nameDest* as it is not.

In the credit card data set, no invalid values were found either and all columns were numbers. Then, no preprocessing was necessary.

In the audit data set, one invalid value was found in the column *Money_Value*. I chose to replace this value by a zero. Furthermore, the column *LOCATION_ID* contains strings. I then decided to remove it. Indeed, the location is not relevant for the study there is then no need to do one-hot encoding for it.

As the financial data set was quite big, I made a first implementation on smaller samples of it, randomly selected. Indeed, I made samples of 200,000 transactions with respect to the ratio of outliers in the initial data set: 155 fraudulent transactions from the initial set and 199,845 regular ones. This first work on samples allowed me to find the best inputs and arguments for every model in order to run them accurately and fast enough on the big original data set. For the two other sets, no first implementation on samples was needed.

First, I implemented the two naive methods: the clustering and the statistical methods.

For the clustering method, I chose to work with the HDBSCAN algorithm as we saw in Chapter 2 that it was the best option as long as we don't know how the data set behaves towards clustering. In Python, an entire HDBSCAN library is available and I decided to use it in my program.

The challenge in the clustering method is to find the minimal size of the clusters that allows the best classification matrix *i.e.* precision, recall, and F1-score (these topics will be developed below). As explained earlier, I first took a random sample to train the HDBSCAN algorithm for the financial data set and tried different cluster sizes. It appeared that for a size of 100 and above I got the best results. Then, I assumed that the minimal size would be proportional to the size of the data set. This led me to having the minimal size of 3000 on the original set. As the credit card data set is approximately of the same size as the sample, I decided to take a size of cluster of 300. And, for the audit set, which is a smaller and more balanced set, I got the best results with a cluster size of 230.

For the statistical method, the first step is to identify which features are the most correlated with the key column, *i.e.* the column labeling if the row is a outlier or not. To find out, I plot the correlation chart as we will see later on. Then, I created a list for every feature labeling 1 or 0 for every row following the absolute mean deviation method described in Chapter 2. Finally, I summed up all these lists to get my raw prediction list.

There are then two methods:

1. The union method: if the number in the raw prediction list is one or more, it is labeled as an outlier in the final prediction list;

2. The intersection method: the strict intersection method will be that if the number in the raw prediction list is equal to the number of features chosen then it is labeled as an outlier. However, to find a balance between this strict method and the union method, I decided to run an alternative intersection method, where I can set up the number starting from which the row is set up as an outlier.

Eventually, the Miller value has to be set. I decided to set it on 2.5 for all sets.

Then, I implemented machine-learning methods. In Python, the *scikit-learn* library provides an implementation of numerous machine learning algorithms. Isolation forest, local outlier factor, logistic regression, and support vector machine are in this library. Therefore, I decided to use it to implement my machine-learning models.

First, to implement machine learning methods, the data sets have to be split in two parts:

- **X**: Each data set without the column:
  - *isFraud* for the financial data set;
  - *Class* for the credit card data set;
  - *Risk* for the audit data set.
- **y**: The column:
  - *isFraud* for the financial data set;
  - *Class* for the credit card data set;
  - *Risk* for the audit data set.

Then, I used *train_test_split* from the *scikit-learn* library. This tool splits arrays or matrices into random train and test subset. It also gives the opportunity for the user to decide the size of the subsets. This preparatory work is common to almost all machine learning methods that need training. I decided to set up the test size to 20% of each set.

To implement the isolation forest, I used the algorithm of *scikit-learn*. All its inputs are optional, but I decided to set some values myself as the results without inputs was not good enough:

- The number of samples to draw from **X** to train each base estimator is set on the length of the training subset of X;
- The contamination is set on 0.0013 for the financial set, 0.0017 for the credit card set, and 0.64 for the audit set. Indeed, it represents the proportion of outliers in the set.

Local outlier factor (LOF) is an unsupervised learning algorithm also implemented in *scikit-learn*. Unlike isolation forest, the data set should not be split into training and test subsets. For this algorithm, I used *fit_predict* from the same library. This tool is useful for unsupervised methods as we will just obtain the labeling results of running our model on our data.

Like isolation forest, LOF has optional inputs that I tried to change in order to get better results:

- The contamination is set to the same values as above except for the audit set, where it is not precised because the contamination has to be within 0 and 0.5 for LOF;
- The number of neighbors to use for k-neighbors is set to 5 for the financial and credit card data set, and 200 for the audit data set.

Support vector machine (SVM) is an algorithm with a great complexity. Therefore, I decided to first test logistic regression on the random samples before trying to run SVM for the financial data set.

From the same *scikit-learn* library, I used the implementation of logistic regression and I used it for this model. As before, it has some default values for the optional parameters, but I decided to set up some values. In this case the values are common to all data sets:

- the inverse of regularization strength value is set on 0.001;
- the penalty is set on *l2*;
- the maximum number of iterations is set on 1000000 in order to be sure that the algorithm converges.

All the other inputs are set to their default values. This algorithm is quite fast and will be preferred to SVM in the case it is too slow.

In the case of SVM, I decided to use two different methods:

- LinearSVC, in which I just set one input the maximum number of iteration on 1,000,000;
- svm.SVC, in which I set the kernel to *poly* with degree to 2 and gamma to *auto*.

These values are again the same for all the sets.

Multi layer perceptron classifier (MLP) was also already implemented in the *scikit-learn* librairy. And I decided not to change any of the default values except the number of maximum iteration, which I set up to 500 for both the credit card and audit set. This number was enough for the audit set however for the other it made the algorithm stop before it actually converges. To address this problem, I should have set a larger number for the credit card set. Unfortunately, my computer was not powerful enough to support it. Similarly, I was not able to run this algorithm with the financial data set because of this problem.

Furthermore, to visualize the performances of my algorithms, I decided to use the classification report from *scikit-learn*. This tool returns a text summary of the accuracy, precision, recall, and F-1 score for each class. Here the classes are 1 and 0 for outliers and inliers. It also returns averages including macro average and weighted average.

To understand all scores returned by the classification report, I must first introduce four properties to define the result of a classification on a data point:

- True positive (TP): if the instance is positive and classified as positive;
- False positive (FP): if the instance is negative but classified as positive;
- True negative (TN): if the instance is negative and classified as negative;
- False negative (FN): if the instance is positive but classified as negative.

The accuracy is calculated as:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.1}$$

However, in cases where one class is way bigger than the other, accuracy is not really relevant.

Recall determines if the number of false negative is high which is important in fraud detection and is calculated as:

$$recall = \frac{TP}{TP + FN} \tag{3.2}$$

Precision determines if the number of false positive is high and is calculated as:

$$precision = \frac{TP}{TP + FP} \tag{3.3}$$

F-1 score is the harmonic mean of precision and recall and is important for uneven class distributions. It is calculated as:

$$F - measure = \frac{2 * recall * precision}{recall + precision} \tag{3.4}$$

3.2 Financial data set

3.2.1 Data description

The financial data set I chose is quite large and complete: It consists of six millions rows. It represents a set of transactions an institution like a bank could have in a week or in a month for example. Indeed, there are several customers initiating and receiving transactions.

It is described as follows:

***step*:** unit of time one step is one hour;

***type*:** type of transaction, there are five different types cash-in, cash-out, debit, payment, and transfer;

***amount*:** amount of money of the transaction;

***nameOrig*:** name of the customer that started the transaction;

***oldbalanceOrig*:** initial balance before the transaction;

***newbalanceOrig*:** new balance after the transaction;

***nameDest*:** name of the customer who received the money;

***oldbalanceDest*:** initial balance of the recipient;

***newbalanceDest*:** new balance of the recipient;

***IsFraud*:** 1 if it is labeled as a fraud, 0 if not;

***IsFlaggedFraud*:** 1 if the transaction is superior or equal to 200 000, 0 if not.

This financial set is saved as a csv file and Figure 3.1 shows the first five rows of the original set before preprocessing. As we can see the entries are of different types:

- *step*, *IsFraud*, and *IsFlaggedFraud* are integers;
- *amount*, *oldbalanceOrig*, *newbalanceOrig*, and *oldbalanceDest* are floats;
- *nameOrig*, and *nameDest* are strings;

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Figure 3.1: First five rows of the data set

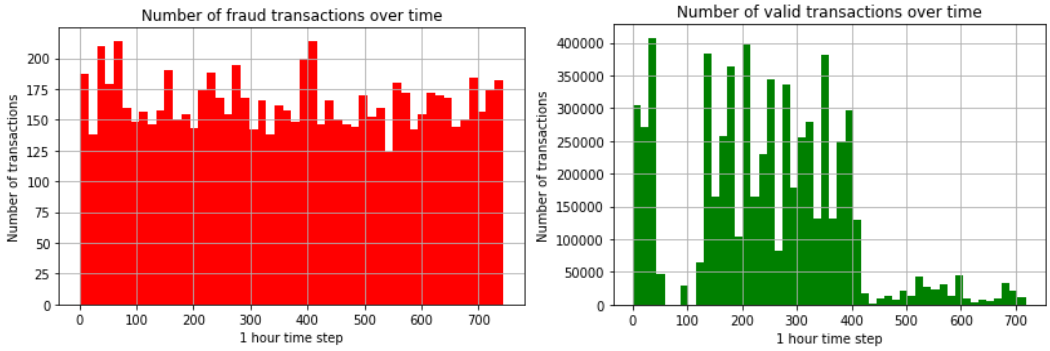


Figure 3.2: Comparison between the number of fraud and valid transactions over time

- *type* will be converted from string to integer via one-hot encoding.

The strings will not be used in the algorithms I chose which are mainly based on mathematical methods. Therefore, I did a preparatory work on the data set to make it fit the algorithms prerequisites as explained above. Besides, I got some information and charts about the set in order to understand better the data and how each category is linked to the others.

First of all, the important data is the column *IsFraud*. Indeed, it will be the key to our algorithm to determine whether a transaction is a fraud (an outlier) or not. In the data set, the fraudulent transactions occur when the *type* is CASH_OUT or TRANSFER. However, it is not correlated with the number of transaction in these *types*. Indeed, there are much more transactions made by PAYMENT than by TRANSFER. In total, there are 8213 frauds for over six millions transactions, with approximately half CASH_OUT and half TRANSFER. The ratio of fraudulent transactions over the total amount of transactions is then quite small: 0.13%.

The number of transactions made over time is also a good indicator to detect a fraud. Indeed, as seen before, there is a remarkable difference between the number of fraudulent transactions and the number of regular ones. Figure 3.2 shows the number of fraudulent and valid transactions over time.

One can notice that the number of fraudulent transactions tends to stay between 125 and 225 per one hour time step, whereas the number of valid transactions is zero at certain time steps and reaches 400,000 at others. This regularity in the distribution of the fraudulent transaction means that the column *step* will not be of great importance when looking for outliers. Moreover, on a more practical side it also means that there are higher chances of fraud at certain times of the day, where transactions are not usually made; for instance at night.

The last tool that might be helpful is the correlation matrix or correlation chart. It will enable the analyst to understand better how all data interact with one another and for the

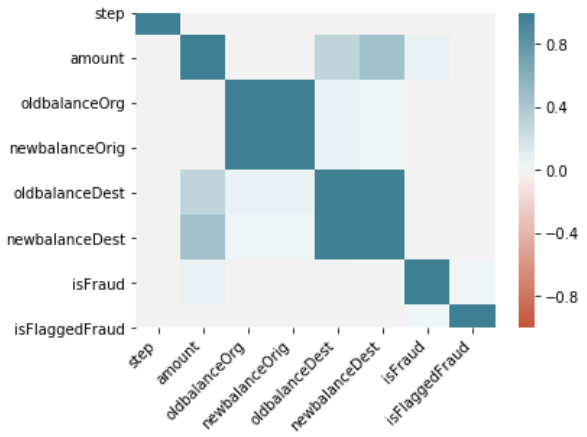


Figure 3.3: Correlation chart of the data set

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	1270904
Class 1	0.09	0.10	0.10	1620
Accuracy			1.00	1272524
Macro avg	0.55	0.55	0.55	1272524
Weighted avg	1.00	1.00	1.00	1272524
Running time				4min31

Table 3.1: Using isolation forest method

statistical method. It is shown in Figure 3.3.

The color bar on the right side of the correlation chart is there to determine where the correlation is maximum; for instance, on the diagonal the value will be one. Similarly, the balance before transaction is highly correlated to the balance after transaction for the payer and the recipient. Unfortunately, this chart is not helpful in our case as nothing is highly correlated to *IsFraud*. Indeed, the *amount* is slightly correlated but nothing remarkable enough to focus my work on. Besides, the *step* and *amount* are not relevant correlated features. That is why I can not do the statistical method on this data set.

3.2.2 Results

Tables 3.1, 3.2, 3.3, and 3.4 summarize the results from all methods for this financial data set.

3.3 Credit card data set

3.3.1 Data description

The credit card data set I chose is approximately 200,000 rows. It is an actual Europeans card holder transactions set. But we can also see this set as transactions one can have on his/her bank accounts in a year for example.

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	1270904
Class 1	0.09	0.10	0.10	1620
Accuracy			1.00	1272524
Macro avg	0.55	0.55	0.55	1272524
Weighted avg	1.00	1.00	1.00	1272524
Running time				9min36

Table 3.2: Using local outlier factor method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	1270904
Class 1	0.36	0.41	0.38	1620
Accuracy			1.00	1272524
Macro avg	0.68	0.71	0.69	1272524
Weighted avg	1.00	1.00	1.00	1272524
Running time				1min42

Table 3.3: Using logistic regression method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	1270904
Class 1	0.00	0.00	0.00	1620
Accuracy			1.00	1272524
Macro avg	0.50	0.50	0.50	1272524
Weighted avg	1.00	1.00	1.00	1272524
Running time				40min15

Table 3.4: Using support vector machine method

It is described as follows:

- *time*: number of seconds between this transaction and the first one;
- *V1* to *V28*: sensitive features;
- *amount*: transaction amount;
- *class*: 1 if it is labeled as a fraud, 0 if not.

The features *V1* to *V28* are sensitive features transformed by the data set maker. They are a result of a principal component analysis (PCA) which is a dimensionality reduction in order to protect users identity.

This set is composed of only numerical values:

- *time*, and *class* are integers;

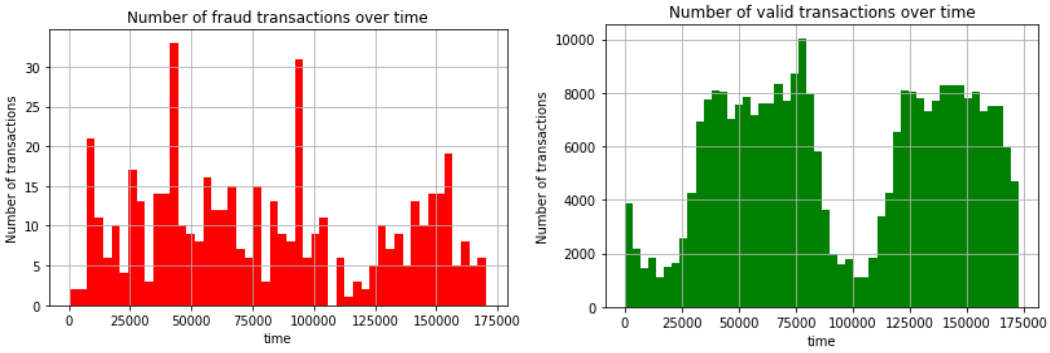


Figure 3.4: Comparison between the number of fraud and valid transactions over time

- *V1* to *V28*, and *amount* are floats.

This makes the preprocessing of data easier as no column has to be removed to apply the algorithms.

As for the financial data set, a good understanding of the set is important to anticipate how the algorithms will behave.

First, the important information in this set is the column *class*. Indeed, it will be the key column for the machine learning algorithms in this case. As in the previous set, this one is very unbalanced. It has 284,315 regular transactions for 492 fraudulent ones. This makes a ratio of fraud of 0.17%. The features amount *V1* to *V28* are within -500 and 500 as we can expect everyday transactions to be.

Then, I plot the number of fraudulent and valid transactions over time to see if I could get any information. These charts are presented in Figure 3.4. Like for the financial data set, it is visible on the valid transaction histogram that transactions are not made with regularity throughout time. Indeed, there are moments of peaks where until 10,000 transactions are made and then, decreases where only 1000 are made. However, there is no regularity in the histogram of fraudulent transactions either.

Finally, I plot a correlation chart of all features to see if some were correlated. It is presented in Figure 3.5. As we can see, none of the feature *V1* to *V28* are correlated to each other, which is normal with a PCA. The feature *V7* and *V20* are positively correlated to the *amount* which means that they make the biggest contribution to it. It is also interesting to see which features are positively correlated to the *class* for the statistical method. Figure 3.6 shows it clearly. I then decided to use the features *V2*, *V4*, and *V11* to proceed with the statistical method. I decided to take the intersection method for this set.

3.3.2 Results

Tables 3.5, 3.6, 3.7, 3.8, 3.9, and 3.10 summarize the results from all methods for the credit card data set.

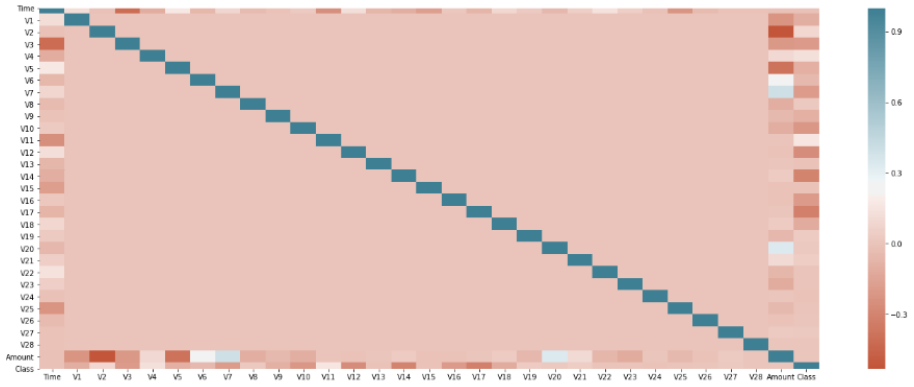


Figure 3.5: Correlation chart of the data set

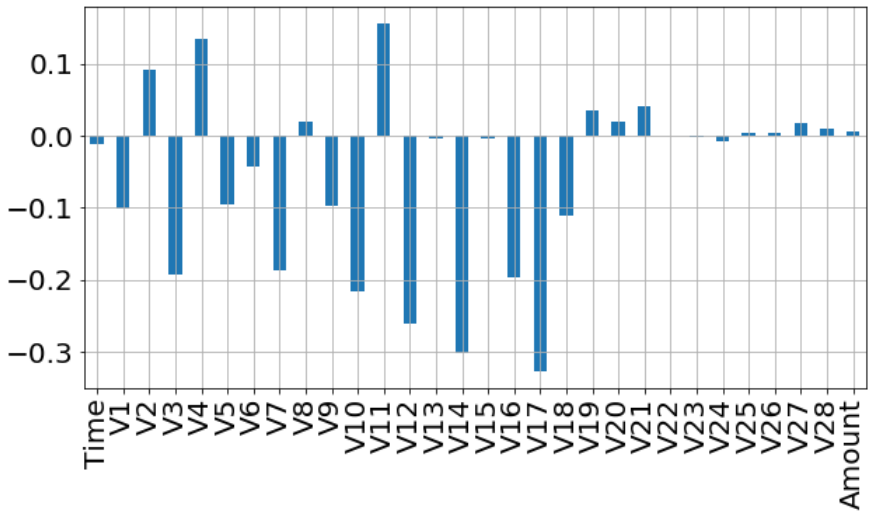


Figure 3.6: Correlation chart of the features towards the *Class*

	Precision	Recall	F-1 score	Support
Class 0	1.00	0.90	0.95	284315
Class 1	0.00	0.12	0.00	492
Accuracy			0.90	284807
Macro avg	0.50	0.51	0.48	284807
Weighted avg	1.00	0.90	0.95	284807
Running time				40min45

Table 3.5: Using the HDBSCAN method

3.4 Audit data set

3.4.1 Data description

To complete this study of fraud detection, I decided to add an audit data set to it. Indeed, fraud detection is also detecting fraudulent firms committing financial crimes. This set is

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	56861
Class 1	0.32	0.34	0.33	101
Accuracy			1.00	56962
Macro avg	0.66	0.67	0.66	56962
Weighted avg	1.00	1.00	1.00	56962
Running time				3min57

Table 3.6: Using isolation forest method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	284315
Class 1	0.11	0.11	0.11	492
Accuracy			1.00	284807
Macro avg	0.55	0.55	0.55	284807
Weighted avg	1.00	1.00	1.00	284807
Running time				3min8

Table 3.7: Using local outlier factor method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	56861
Class 1	0.89	0.57	0.70	101
Accuracy			1.00	56862
Macro avg	0.95	0.79	0.85	56962
Weighted avg	1.00	1.00	1.00	56962
Running time				9min20

Table 3.8: Using logistic regression method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	56861
Class 1	0.00	0.00	0.00	101
Accuracy			1.00	56962
Macro avg	0.50	0.50	0.50	56962
Weighted avg	1.00	1.00	1.00	56962
Running time				

Table 3.9: Using support vector machine method

smaller than the others, approximately 800 rows, each of them representing a firm.

It is described as follows:

- *Sector_score*: historical risk score value of the target unit;

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	284315
Class 1	0.59	0.49	0.53	492
Accuracy			1.00	284807
Macro avg	0.80	0.74	0.77	284807
Weighted avg	1.00	1.00	1.00	284807
Running time				34.6s

Table 3.10: Using statistical method

	Precision	Recall	F-1 score	Support
Class 0	1.00	1.00	1.00	56861
Class 1	0.65	0.83	0.73	101
Accuracy			1.00	56962
Macro avg	0.83	0.92	0.86	56962
Weighted avg	1.00	1.00	1.00	56962
Running time				28min3

Table 3.11: Using multi layer perceptron method

- *LOCATION_ID*: unique ID of the city/province;
- *PARA_A*, *PARA_B*: discrepancy found in the planned expenditure of inspection;
- *Total*: sum of *PARA_A* and *PARA_B*;
- *score_A*, *score_B*, *Score_B.1 score_MV*, *Score*: historical risk scores of the target unit;
- *History*: average historical loss suffered by the firm in the last 10 years;
- *Money_Value*: amount of money involved in misstatements in the past audits;
- *District_Loss*: amount of loss suffered by the firm in the last year;
- *Numbers*: historical discrepancy score;
- *Risk_A*, *Risk_B*, *Risk_C*, *Risk_D*, *Risk_E*, *Risk_F*: risk class assigned to an audit case;
- *Audit_Risk*: total risk score using analytical procedure;
- *Inherent_Risk*, *CONTROL_RISK*, *Detection_Risk*: other risks related to the firms;
- *Prob*, *PROB*: probabilities related to the firm.

The features used to describe the firms are economic features that need a certain level of knowledge to understand but it is not the point of this study. In this set, every data is an integer or a float except for the column *LOCATION_ID* which has strings. Therefore, and because it is of no use for my study I decided to remove this column and not to do one-hot

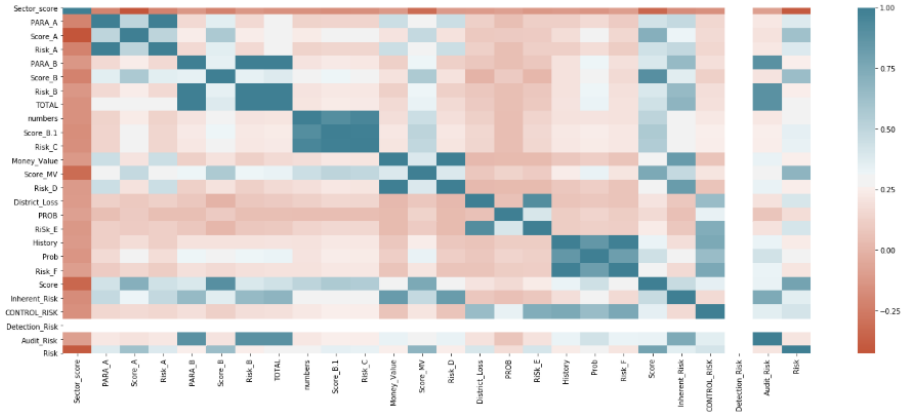


Figure 3.7: Correlation chart of the data set

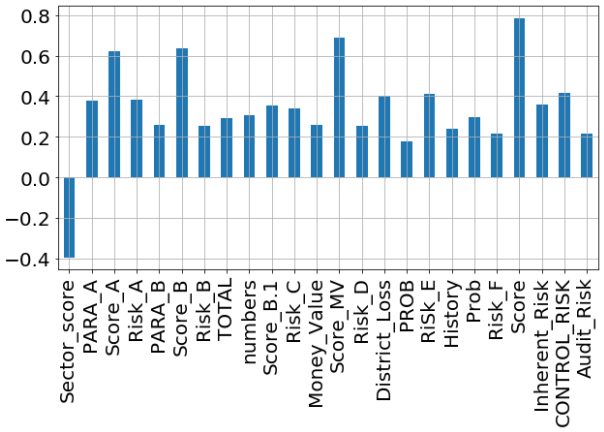


Figure 3.8: Correlation chart of the features towards the *Risk*

encoding. Also, a value was missing in the column *Money_Value* and I decided to replace it by 0.

This set, unlike the others is quite balanced. It has 305 fraudulent firms and 471 regular ones. The score features are within 0 and 10, whereas the others take different values.

A correlation chart presented on Figure 3.7 shows the correlation between all features. It is visible here that the column *Detection_Risk* is of no use. Indeed, it has just one same value for all firms: 0.5. It can be seen on this chart that all features have a positive correlation with the *Risk* except the *Sector_score*. For the statistical method, the features I chose to take are *Score_A*, *Score_B*, *Score_MV*, and *Score*. For this data set, I chose the intersection method but with only half of the features necessary to label a row as an outlier.

3.4.2 Results

Tables 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, and 3.18 summarize the results from all methods for the audit data set.

	Precision	Recall	F-1 score	Support
Class 0	0.67	1.00	0.81	471
Class 1	1.00	0.26	0.41	305
Accuracy			0.71	776
Macro avg	0.84	0.63	0.61	776
Weighted avg	0.8	0.71	0.65	776
Running time				429ms

Table 3.12: Using the HDBSCAN method

	Precision	Recall	F-1 score	Support
Class 0	1.00	0.65	0.79	101
Class 1	0.61	1.00	0.76	55
Accuracy			0.78	156
Macro avg	0.81	0.83	0.77	156
Weighted avg	0.86	0.78	0.78	156
Running time				1s13

Table 3.13: Using isolation forest method

	Precision	Recall	F-1 score	Support
Class 0	0.90	0.88	0.89	471
Class 1	0.82	0.86	0.84	305
Accuracy			0.87	776
Macro avg	0.86	0.87	0.87	776
Weighted avg	0.87	0.87	0.87	776
Running time				82ms

Table 3.14: Using local outlier factor method

	Precision	Recall	F-1 score	Support
Class 0	0.93	1.00	0.96	101
Class 1	1.00	0.85	0.92	55
Accuracy			0.95	156
Macro avg	0.96	0.93	0.94	156
Weighted avg	0.95	0.95	0.95	156
Running time				24.3s

Table 3.15: Using logistic regression method

	Precision	Recall	F-1 score	Support
Class 0	0.65	1.00	0.79	101
Class 1	0.00	0.00	0.00	55
Accuracy			0.65	156
Macro avg	0.32	0.50	0.39	156
Weighted avg	0.42	0.65	0.51	156
Running time				53.4ms

Table 3.16: Using support vector machine method

	Precision	Recall	F-1 score	Support
Class 0	0.89	0.95	0.92	471
Class 1	0.91	0.81	0.86	305
Accuracy			0.89	776
Macro avg	0.90	0.88	0.89	776
Weighted avg	0.90	0.89	0.89	776
Running time				130ms

Table 3.17: Using statistical method

	Precision	Recall	F-1 score	Support
Class 0	0.99	1.00	1.00	101
Class 1	1.00	0.98	0.99	55
Accuracy			0.99	156
Macro avg	1.00	0.99	0.99	156
Weighted avg	0.99	0.99	0.99	156
Running time				40.1s

Table 3.18: Using multi layer perceptron method

Chapter 4
Analysis and discussion

Now that I have presented the raw results in Chapter 3, I am going to present in this section previous experiments and results obtained with the three same data sets that I have worked on. The aim is to see if the work done in this thesis obtains better results. I will then detail a little more the results obtained in Chapter 3 and discuss their relevance with regard to previous works. Finally, I will propose some possible improvements for fraud detection in these types of data sets.

4.1 Related works

First, several articles and projects mention the financial data set. Indeed, as no financial set is available online due to privacy issues, a synthetic financial data set like this one is valuable when it comes to experimenting on fraud detection. Among the works I have seen on these data sets, most of them use techniques I did not use in this thesis. To manage a large data set like this one, some decided to work only on a random sample of 100,000 rows or to use sampling techniques.

Besenbruch (2018), for instance, decided to perform data reduction, sampling, and classification on the set. For this, Besenbruch used data reduction algorithms and sampling techniques such as random over sampling, random under sampling, and SMOTE. After this data preprocessing, three algorithms were run: random forest, gradient boosting, and logistic regression. However he detailed results of all his tests: with and without set reduction. With normalized data and no reduction, he achieved a F-1 score of 0.68 for logistic regression, 0.90 for random forest, and 0.75 for gradient boosting. With standardized data and no reduction, the F-1 scores for the same algorithms are respectively 0.78, 0.90, and 0.78. These results are higher than the ones I have obtained but the financial set has been changed and transformed beyond what I did in this thesis. Indeed, my aim was to test algorithms on data as close to reality as possible. Then, with all the sampling methods his results are even higher than the ones above. There is however no need to discuss them as the set is even more transformed

and his method and approach was very different from mine.

Kaur (2019) had a similar approach than the one I had. He decided to keep all the data set to train his algorithms which are logistic regression, naive Bayes, random forest, and extreme gradient boosting. He has achieved an F-1 score for outliers *i.e.* class 1 of 0.48 for logistic regression, 0.14 for naive Bayes, 0.87 for random forest, and 0.92 for extreme gradient boosting. These results, except the naive Bayes method also have results better than the ones I have. However he did not mention the running time in his studies though it would have been interesting to compare.

Vu (2020) has also worked on this data set. However, he published his results only with accuracy scores. Besides, he decided to work with a 100,000 rows sample of the set. As my accuracy scores are 1.00 for every algorithms as the set is very unbalanced, my results were better than his. But, I do not consider accuracy score to be a relevant element of comparison in the case of this data set. Furthermore, the fact that he has decided to conduct his study with a sample of the set is also not relevant to me as the challenge of this set was to run algorithms able to manage a six million rows set.

Then, less papers and works used the credit card data set to detect fraud. The main work achieved with this set is the accuracy in creating a random credit card transactions set taking it as a model. Besides, the fraud related works on this data set used deep learning and neural networks to detect fraudulent transactions.

Sabari (2020), for example, decided to use deep learning to detect fraud. He especially used convolutional neural networks (CNN), random forest classifier, and multilayer perceptron classifier (MLP classifier) with different parameters. All of these being neural networks algorithms far more powerful than the machine learning algorithms I used in this thesis. Indeed, he managed to achieve an average F-1 score of 0.86 for CNN, 0.84 for random forest classifier, 0.81 for the first MLP classifier and 0.71 for the second one. These F-1 scores are pretty high and they are the best results obtained for this data set. Unfortunately, it is not really relevant to compare to my work as I did not go that far in my fraud detection algorithms.

Finally, two articles related results concerning the audit data set. As the set is pretty well balanced it is relevant to look at the F-1 score and the accuracy.

Tiwari and Hooda (2018) tested ten different machine learning methods with this data set: decision tree, adaboost, random forest, support vector machine, logistic, decision stump, J48, naive Bayes, Bayesian network, and neural network. The best accuracy score is 0.94 and is obtained with random forest. The best F-1 score is 0.93 and is also obtained with random forest. These experiments are based on the principle of ten fold cross validation technique. The data set is split into ten equal subsets and the algorithms train on 9 of these and test on the last one. This is repeated a certain amount of time and then the results are summarized.

Similarly, Hooda et al. (2018) tested ten machine learning techniques on this data set: decision tree, adaboost, random forest, support vector machine, probit linear model, neural network, decision stump model, J48, naive Bayesian, and Bayesian network. Here, they decided to keep only nine features: *PARA_A*, *PARA_B*, *Total*, *Money_Value*, *Numbers*, *Score_MV*, *History*, *District_Loss*, and *Score*. It is out of these features only that experiments have been conducted. They then achieved the best accuracy with J48 and Bayesian network with a score of 0.94 and the best F-1 score with the same two methods with a score of 0.95. These experiments were also held with the ten fold cross validation technique. Similarly, these results are less relevant as they do not take the set with all its features.

	Financial set	**Credit card set**	**Audit set**		
	F-1 score (Class 1)	F-1 score (Class 1)	F-1 score (Class 0)	F-1 score (Class 1)	Accuracy
HDBSCAN		0.00	0.81	0.41	0.71
Statistical method		0.53	0.92	0.86	0.89
Isolation forest	0.10	0.33	0.79	0.76	0.78
Local outlier factor	0.10	0.11	0.89	0.84	0.87
Logistic regression	**0.38**	**0.70**	**0.96**	**0.92**	**0.95**
Multi layer perceptron		0.73	1.00	0.99	0.99
Average running time	14min	11min		13s	

Table 4.1: Results

4.2 Data sets and models

In this thesis, I chose to work with three different data sets. The aim was to see if outlier detection algorithms could work on all types of fraud detection. Indeed, financial fraud can occur in banks, in personal credit card transactions, or in audit of enterprises. These three different fields have different data sets which implies different features and different size. Therefore, it is a good way to test algorithms and compute their efficiencies. That is why I decided to take running time into account in my study.

This thesis was also conducted with the same algorithms on the three different sets in order to see if one stood out from the others or if every set had its own best algorithm. I decided to work with HDBSCAN, a clustering algorithm, a statistical algorithm I implemented, and four machine learning algorithms from *scikit-learn* isolation forest, local outlier factor, logistic regression, and support vector machine. Plus to go a bit further, I added a neural network algorithm to my comparison with multi layer perceptron. I detailed these methods theoretically in Chapter 2 and practically in Chapter 3. Indeed, I made several tests and looked up for the best choice of parameters before running my algorithms with the three data sets I chose. For the statistical method for instance, I decided not to use the exact same technique with the banking set and with the audit set. Indeed, I wanted to show here the best results one can achieve when using one statistical method.

When looking among all results, we observe that firstly the support vector machine algorithm does not work for any of the three sets. Indeed, due to a convergence problem it gives a result where all rows are considered inliers and none are labeled as outliers. Therefore I will not take it into account in the discussion of the efficiency of algorithms. Secondly, it appears that precision, recall, and F-1 score for class 0 is 1.00 with every algorithms for the two first sets and so is the accuracy.

To have a better view of all results and to help in the comparison and discussion, I decided to summarize the results for all algorithms and all sets in Table 4.1. As explained above I will not take into account the SVM algorithm in this table nor the class 0 F-1 score for the financial and the credit card data set. However, as the audit data set was balanced I decided to take into account the accuracy.

The best results in Table 4.1 are in bold. As we can see, logistic regression algorithm seems to be giving the best results for all three sets. Indeed, I do not consider here the deep learning

algorithm as my study was focused on the improvement between naive and machine learning methods. The results with multi layer perceptron are just to give an idea of what can still be made beyond machine learning methods.

The results for the financial set are really low compared to what has been done in the previous work on this set. Indeed, the best result on this data was obtained by Kaur with an F-1 score for class 1 of 0.92. My best result was a score of 0.38. Even with no comparison a score of 0.38 is far from what can be expected from a good algorithm. In this case, the size of the data set might be the reason of the low scores I obtained. Indeed, in the case of the clustering and statistical methods I did not get any results because the set was not made for these models.

For the credit card data set, the results are better. I achieved a score of 0.70. With this set I managed to run all algorithms. However, I got a score of 0.00 for the clustering method, which indicates that this algorithm was not working on this set either. For the statistical method I got 0.53 which is pretty good. Isolation forest, and local outlier factor got low results on this set. As for the financial data set, my results are lower than what was done in previous works. Using deep learning techniques, Sabari got an F-1 score of 0.86.

Finally, concerning the audit data set, I achieved competitive results for almost all algorithms. The clustering method was once again the one that gave the worst results but still with an F-1 score of 0.41 for class 1 and an accuracy of 0.71. The logistic regression algorithm got an accuracy and an average F-1 score of 0.95. This score is better than the one Tiwari and Hooda obtained. Indeed, they got a best F-1 score of 0.93 and a best accuracy of 0.94. Hooda et al. also got a best accuracy of 0.94 when selecting only nine features of the set. With this method, they achieved the same best average F-1 score as me.

Unfortunately, none of the previous work on these data sets mention the running time of the algorithms they used. I actually saw a real difference running the same algorithms on three data sets of different size, especially with the clustering algorithm and support vector machine. In my opinion, this difference is directly due to the size of the sets and the complexity of these algorithms.

To sum up, I did not obtain competitive results for the financial data set. The set was too big for the clustering algorithm and its features did not fit my statistical method. The results I obtained with the credit card data set was quite good but some better results were achieved using deep learning algorithms. And, concerning the audit data set I got the same, and better results than previous works done on the set. Besides, all algorithms I used got pretty good results with this set. Running times were also really good.

4.3 Possible improvements

The algorithms used in this thesis proved that using machine learning methods leads to better results than naive methods. In this case, logistic regression was the algorithm achieving the best results. However, as seen in the previous works using the credit card data set, and with MLP deep learning algorithms seem to have an even better result. Therefore, if we want to have a state of the art business application for fraud detection I think going for neural networks is a good option. Indeed, I achieved better results with multi layer perceptron than with machine learning or naive algorithms. For the audit data set the results were close to perfect.

I decided to focus in this thesis only on machine learning methods. Indeed, as mentioned before companies are still using rule-based techniques. It is therefore relevant to present machine learning techniques as an alternative and deep learning as an improvement to be implemented further on. In businesses where historical strategies prevail, transitions have to be made one step at a time.

Furthermore, if we do not want to go for deep learning the previous works using the financial data set showed that data reduction, sampling, and classification can also be a good way to ameliorate results.

However, in the case of a small and balanced data set (<1000 rows), going with the logistic regression gives good enough results. Especially in small infrastructures, where they do not have the mean to develop large deep learning applications for fraud detection, machine learning methods can be a good option especially when combined with other techniques such as sampling, reduction, or classification. But this has to be further investigated to be sure it can be implemented in a business context.

Chapter 5
Conclusion

The aim of this thesis was to see how outlier detection algorithms work in a financial fraud detection context. Indeed, nowadays most companies use conventional techniques which are based on rules, implemented by hands, and take a large amount of time to run. Knowing all data processed every day by these firms, machine learning techniques appear to be a better option.

In this thesis, I decided to study a financial, a credit card, and an audit data set in order to have a global view on what techniques can be used to detect fraud. These three sets were chosen with different size, features and balance to match real life sets. Then, even if I focused my work on outlier detection methods I chose to work both with naive and machine-learning algorithms to see how much improvement I could get. I selected a clustering method which I ran with HDBSCAN, a statistical method I implemented myself using the mean absolute deviation, and four machine learning algorithms: isolation forest, local outlier factor, logistic regression, and support vector machine all ran with the *scikit-learn* library.

It appeared; as expected; that the naive techniques were the ones giving the worst results with every set. It could even not be implemented for the first set because of its size. And logistic regression was the algorithm giving the best results with the smallest running time for all sets. This proved that outlier detection machine learning techniques are a better alternative than naive ones when dealing with financial fraud detection. Especially when dealing with finance audit of companies where I achieved notably good results, better than the ones of previous works done on this set.

Nevertheless, nowadays most fraudulent transactions occur with credit card due to the development of dematerialized money and e-commerce therefore the second set has a special importance. And, even if the results with machine learning techniques were quite good some even better results were achieved with deep learning techniques. Indeed, deep learning and neural networks seem to be the next step in detecting fraudulent transactions. However, as most companies are still working with conventional techniques which implies an implementation by hand machine learning seems to be the best alternative in the current business context. To a lesser extent, neural networks have a more consequent running time that has

to be taken into account. Indeed, it needs powerful computers that not all companies can afford.

References

Alpaydin, E. (2020). *Introduction To Machine Learning fourth edition*. Massachusetts Institute of Technology press.

Bernstein, J. H. (2009). The data-information-knowledge-wisdom hierarchy and its antithesis. In Jacob, E. K. and Kwasnik, B., editors, *Proceedings North American Symposium on Knowledge Organization*, volume 2, pages 68–75, Syracuse, NY.

Besenbruch, J. (2018). Fraud detection using machine learning techniques. *Research Paper Business Analytics*.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104.

Campello, R. J., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Chuprina, R. (2020). Credit card fraud detetction solutions - how to implement them in your business. *SPD group*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Technical report, AT&T Labs-Research, USA.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD-96 Proceedings*.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hariri, S., Kind, M. C., and Brunner, R. J. (2018). Extended isolation forest. In *IEEE Transactions on Knowledge and Data Engineering*.

Hooda, N., Bawa, S., and Rana, P. S. (2018). Fraudulent firm classification: a case study of an external audit. *Applied Artificial Intelligence*, 32(1):48–64.

Hosmer, D. W., Jovanovic, B., and Lemeshow, S. (1989). Best subsets logistic regression. *Biometrics*, pages 1265–1270.

Ivakhnenko, A. G. and Lapa, V. G. (1967). Cybernetics and forecasting techniques. *American Elsevier Pub. Co.*

Kaur, G. (2019). Development of business intelligence outlier and financial crime analytics system for predicting and managing fraud in financial payment services. Master's thesis, University of Stirling.

Laughlin, P. (2014). Holistic customer insight as an engine of growth. *Journal of Direct, Data and Digital Marketing Practice*, 16(2):75–79.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*.

Lutins, E. (2017). Dbscan: What is it? when to use it? how to use it. *Medium*.

Miller, J. (1991). Reaction time analysis with outlier exclusion: Bias varies with sample size. *The quarterly journal of experimental psychology*, 43(4):907–912.

Sabari, R. (2020). Improving credit card fraud detection based on cnn/gan.

SAS (2020). Fraud detection and machine learning: what you need to know.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.*, 42(3).

Sithic, H. L. and Balasubramanian, T. (2013). Survey of insurance fraud detection using data mining techniques. *arXiv preprint arXiv:1309.0806*.

Srivastava, D. K. and Bhambhu, L. (2005-2009). Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology*.

Sun, L., Versteeg, S., Boztas, S., and Rao, A. (2016). Detecting anomalous user behavior using an extended isolation forest algorithm: an enterprise case study. *arXiv preprint arXiv:1609.06676*.

Tiwari, A. and Hooda, N. (2018). Machine learning framework for audit fraud data prediction.

Vapnik, V. (2006). *Estimation of Dependences Based on Empirical Data*. Springer Verlag, New York, second edition. English translation of the Russian version, Nauka, Moscow, 1979.

Vu, H. (2020). Machine learning in money laundering detection. *Chair's Message*, page 60.

Yiu, T. (2019). Understanding neural networks.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-08-27

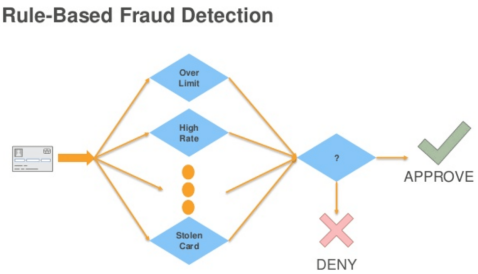
EXAMENSARBETE Outlier Detection Methods Applied to Financial Fraud
STUDENT Églantine Boucher
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Jörn Janneck (LTH)

Outlier detection methods to detect fraud

POPULÄRVETENSKAPLIG SAMMANFATTNING **Églantine Boucher**

Fraud is a challenge our banking system has to face everyday. Most of the time, fraudulent transactions can be modeled as statistical outliers and thus can be found efficiently with outlier detection algorithms. Indeed, combined with machine learning, they give remarkable results

Currently, to prevent fraudulent transactions, financial organizations use rule-based algorithms (see figure). However, in general, machine-learning methods are known to achieve better results in terms of speed and accuracy.



In this thesis, I implemented two naive, four machine-learning, and one deep learning methods on three different sets (the implementation uses Python to run them all). The three sets represent: a statement of a bank, an individual account statement, and an audit on several firms. The aim was to run all the algorithms on every set in order to compare their efficiencies on a large scale.

The three datasets are annotated making the use of supervised learning possible. However, in real life, it is not always the case. It is then relevant to see how unsupervised algorithms behave. To address this problem, the thesis evaluates models pertaining to both supervised and unsupervised learning. Similarly, I implemented a clustering and a statistical method to see how much improvement machine learning methods can achieve compared with naive ones.

The F-1 score is the evaluation metric To compare the algorithms. This score is defined as the harmonic mean of the precision (ratio of true positives on all positives instances) and recall (ratio of true positives on true positives and false negatives).

The results were quite the same for the two statement sets as they were both uneven (less than 1% fraudulent transactions) and large (over six million, resp. three hundred thousand rows). These results were better for the audit set. More precisely, this work achieved an improvement of 32% for the individual statement set, and of 45% for the audit set when it compared naive to machine learning methods (especially logistic regression). For the bank statement set however, naive methods could not be implemented due to the size of the set. This proves our point that machine learning outlier detection methods are a better way to prevent fraud than naive ones.

Moreover, If we want to get even better results on larger sets, deep learning and neural networks seem to be a good option.