University of Salerno

Dipartimento di Scienze Economiche e Statistiche

PhD in Economics and Policy Analysis of Market and Firms

Curriculum in

Statistical Methods

Anomalies detection in credit risk data: an approach based on the Isolation Forest

Supervisor
Prof. M. Niglio
Coordinator
Prof. S.P. Destefanis

Student
Fabio Forte

XXXI Cicle

Contents

List of Figures

Introduction

Risk can be referred as the chances of having an unexpected or negative outcome. Any action or activity that leads to loss of any type can be termed as risk. Financial institutions such as banks hold typically thousands of financial positions. In addition to the aggregated market and credit risk of these positions, an institution is exposed to other risks which must be measured and managed. Widely, risks can be classified into three types:

1. **Business Risk:** These types of risks are taken by business enterprises themselves in order to maximize shareholder value and profits. As for example: Companies undertake high cost risks in marketing to launch new product in order to gain higher sales;

2. **Non-Business Risk:** These types of risks are not under the control of firms. Risks that arise out of political and economic imbalances can be termed as non-business risk;

3. **Financial Risk:** Financial Risk as the term suggests is the risk that involves financial loss to firms. Financial risk generally arises due to instability and losses in the financial market caused by movements in stock prices, currencies, interest rates and more.

Financial risk is one of the high-priority risk types for every business. **Financial risk** involves financial transactions such us company loans, and its exposure to loan default. The term is typically used to reflect an investor's uncertainty of collecting returns and the potential for monetary loss. For example a business takes a financial risk when it provides financing of purchases to its customers, due to the possibility that a customer may default on payment.

Chapter 1

Introduction to Risk

1.1 Multiple perspectives of Risk

Within risk categories shown in the introduction paragraph a bunch of different risks can be identified:

- **Credit risk.** Credit risk is the risk associated with a borrower going into default. In other words the borrower is not respecting the repayment schedule. Investor losses include lost principal and interest, decreased cash flow, and increased collection costs. A company must handle its own credit obligations by ensuring that it always has sufficient cash flow to meet the repayment schedule. Otherwise, suppliers may either stop extending credit to the company, or even stop doing business with it. Losses can arise in a number of different circumstances:

  - A company does not pay an employee's earned wages when due;
  - A consumer may fail to make a payment due on a mortgage loan, credit card, line of credit, or other loan;
  - A company or government bond issuer does not make a payment on a coupon or principal payment when due;
  - An insolvent bank won't return funds to a depositor.

  From these examples, it is easy to identify the main sub-components of the credit risk:

  a. **Country Risk.** Country risk refers to the risk that a country won't be able to honor its financial commitments. When a country defaults on its obligations, this can harm the performance of

all other financial instruments in that country as well as other countries it has relations with. Country risk applies to stocks, bonds, mutual funds, options and futures that are issued within a particular country. This type of risk is most often seen in emerging markets or countries which have a severe deficit in the legal environment, high levels of corruption, and a low score for most of the socioeconomic variables such as income disparity;

b. **Sovereign Risk.** Many analysts use sovereign debt ratings as a proxy for country risk despite the fact that credit rating agencies do not intend their sovereign debt ratings to speak to country risk because these two risk sources are conceptually distinct. Indeed the sovereign ratings capture the risk of a country defaulting on its commercial debt obligations while the country risk takes also in account the downside of a country's business environment;

c. **Counterparty Credit Risk.** Last but not least, counterparty risk is the risk to each party of a contract that the counterparty will not live up to its contractual obligations. Counterparty risk is a risk to both parties and should be considered when evaluating a contract.

- **Market risk.** Market risk can be defined as the risk of losses in on and off-balance sheet positions arising from adverse movements in market prices.
  This kind of risk is also known as Systematic or un-diversifiable risk because the uncertainty inherent to the entire market or entire market segment.
  Interest rates, recession and wars all represent sources of systematic risk because they affect the entire market and cannot be avoided through diversification. Systematic risk can be mitigated only by being hedged.
  For a bond for example, market risk arises from the variability of interest rates, this variability is referred to as volatility.
  Volatility is a measure of risk because it refers to the behavior of the investment rather than the reason for this behavior. Market movements enable people to make money from stocks, volatility is essential for returns, and the more unstable the investment the more chance there is that it will experience a dramatic change in either direction.
  β is a measure of the volatility [66], or systematic risk, of a security or a portfolio in comparison to the market as a whole. In other words, β gives a sense of a stock's market risk compared to the greater market

and enables to compare a stock's market risk to that of other stocks. This measure of the volatility is used in the capital asset pricing model (CAPM) introduced by Sharpe in [66].
β is calculated using regression analysis, and you can think of beta as the tendency of a security's returns to respond to swings in the market. β can be interpreted according to the following line of thinking:

1. $\beta > 1$. The security's price will be more volatile than the market, i.e. if a stock's β is 1.2, then, theoretically, the security's price is 20% more volatile than the market;
2. $\beta = 1$. It indicates that the security's price will move with the market;
3. $\beta < 1$. It means that the security will be less volatile than the market;
4. $\beta = 0$. Basically, cash has a β of 0. In other words, regardless of which way the market moves, the value of cash remains unchanged under the assumption of absent inflation;
5. β **is negative.** A β less than 0 indicates an inverse relation to the market which is possible but highly unlikely. Some investors used to believe that gold and gold stocks should have negative β because they tended to do better when the stock market declined.

From a regulatory perspective, market risk stems from all the positions included in banks' trading book as well as from commodity and foreign exchange risk positions in the whole balance sheet. Traditionally, trading book portfolios consisted of liquid positions easy to trade or hedge. However, developments in banks' portfolios have led to an increase in the presence of credit risk and illiquid positions not suited to the original market capital framework.
Market risk can be broken down as follow:

a. **Equity risk.** Equity risk covers the risk involved in the volatile price changes of shares of stock. This risk can be defined as the financial risk involved in holding equity in a particular investment. Besides, equity risk premium is the excess return that an individual stock or the overall stock market provides over a risk-free rate. This excess compensates investors for taking on the relatively higher risk of the equity market. The size of the premium can vary as the risk in the stock, or just the stock market in general, increases. For example, higher risks have a higher

premium. The concept of this is to entice investors to take on riskier investments;

b. **Interest rate risk.** Interest rate risk is the risk that an investment's value will change as a result of a change in interest rates. This risk affects the value of bonds more directly than stocks. In other words, this risk arises for bond owners from fluctuating interest rates. How much interest rate risk a bond has depends on how sensitive its price is to interest rate changes in the market. The sensitivity depends on two things, the bond's time to maturity, and the coupon rate of the bond;

c. **Currency risk.** Currency risk is the risk that foreign exchange rates or the implied volatility will change, which affects, for example, the value of an asset held in that currency. When investing in foreign countries you must consider the fact that currency exchange rates can change the price of the asset as well. Foreign-exchange risk applies to all financial instruments that are in a currency other than your domestic currency;

d. **Commodity risk.** Commodity risk is the risk that commodity prices (e.g. corn, copper, crude oil) or implied volatility will change.

- **Liquidity risk.** Liquidity risk refers to the inability to adjust positions at current market rates or increased funding costs. In other words, it relates to the ability of an economic agent to exchange his or her existing wealth for goods and services or for other assets. Liquidity should be understood in terms of flows (as opposed to stocks), where liquidity refers to the ability of realizing these flows. Inability of doing so would render the financial entity illiquid. The three main types of liquidity risk are well explained by Nikolaou in [59]:

  a. **Central bank liquidity risk.** Central bank liquidity is the ability of the central bank to supply the liquidity needed to the financial system. It is typically measured as the liquidity supplied to the economy by the central bank, i.e. the flow of monetary base from the central bank to the financial system.
  Nevertheless the central bank liquidity risk should not be non-existent, as the central bank is always able to supply base money and, therefore, can never be illiquid. Typically the central bank, being the monopoly provider of liquidity, i.e. the originator of the monetary base, can dispense liquidity as and when it deems

needed, so as to satisfy the equilibrium demand for liquidity in the banking system (avoiding cases of excess liquidity or liquidity deficits).
A central bank can only be illiquid to the extent that there is no demand for domestic currency, and therefore the supply of base money from the central bank could not materialize. This could happen in cases of hyperinflation or an exchange rate crisis;

b. **Funding liquidity risk.** The Basel Committee of Banking supervision defines funding liquidity risk as the ability of banks to meet their liabilities, unwind or settle their positions as they come due [5]. However, references to funding liquidity have also been made from the point of view of traders i.e. Brunnemeier and Pedersen in [19] or investors i.e. Strahan in [68], where funding liquidity relates to their ability to raise funding (capital or cash) in short notice. Namely, an entity is liquid as long as inflows are bigger or at least equal to outflows. Measuring funding liquidity risk is not trivial. In most cases practitioners construct various funding liquidity ratios, which reveal different aspects of the availability of funds within a certain time horizon ahead and use them as proxies for funding liquidity risk. Such measures can be produced either by static balance sheet analysis or by dynamic stress testing techniques and scenario analysis. The latter is more cumbersome to calculate if only because it relies on complicated calculations and a wider set of information and hypotheses. Drehmann and Nikolaou in [27] suggest a simple and more straightforward proxy, based on the role of the central bank as a potential funding liquidity source. They argue that bidding behavior in central bank auctions can reveal the funding liquidity risk of banks over a one week horizon and construct proxies of funding liquidity risk from bidding data;

c. **Market liquidity risk.** The notion of market liquidity has been introduced by Keynes in [48]. A number of more recent studies define market liquidity as the ability to trade an asset at short notice, at low cost and with little impact on its price. Starting with the liquidity-based asset pricing model of Holmstro and Tirole showed in [46], asset pricing models typically measure liquidity risk as the covariance (commonality) between a measure of liquidity (innovations) and market returns.
Brunnemeier and Pedersen in [19] explain that liquidity risk is in

most cases low and stable, it assumes a rare and episodic nature which result from downward liquidity spirals due to mutually reinforcing funding and market illiquidity. In other words the related literature suggests that asset prices reflect liquidity costs, which are linked to the existence of liquidity risk which means that is also possible to predict future returns based on current liquidity risk estimates.
Notably, liquidity risk can lead to financial crises, which damage financial stability, disrupt the allocation of resources and ultimately, affect the real economy. Given the importance of market liquidity risk (i.e. systemic risk) to financial stability, it is the type of liquidity risk that immediately alerts policy makers.

- **Operational risk.** Operational risk corresponds to the possibility of mistakes or technical problems in trading or risk management operations. This includes the mis-understanding of involved risks, the mis-pricing of instruments, fraud, or system failure; In other words, it represents the probability that the value of a financial asset is influenced by unpredictable factors resulting from a company's ordinary business activities.
For example a bank employee can incur into calculation errors, or into procedural blocks that could temporarily prevent the correct execution of financial transactions. As well as of market risk, a proper evaluation of operational risk is provided by advanced statistical models. Among others, the Advanced Measurement Approaches (AMA) has to be mentioned and it is the most diffused model type and bases on the modelling of all the events from which operational risks derive. This is built by collecting frequency data and by considering variables with a predictive power in terms of future incident occurrence. The idea behind is to evaluate the operational risks and include this estimation into the risk weighted asset calculation.
Operational risk categories includes:

  a. **Legal risk.** This kind of operational risk arises out of legal constraints such as lawsuits. Whenever a company needs to face financial loses out of legal proceedings, it is legal risk;

  b. **Fraud Risk.** Fraud Risk identifies and addresses an organization's vulnerability to both internal and external fraud.
  Fraud, by definition, entails intentional misconduct, designed to evade detection. As such, the fraud risk assessment should an-

ticipate the behavior of a potential fraud perpetrator. A correct assessment of the potential fraud ensure organization's ability to maintain operations and reputation.
Therefore, control activities should always consider both the fraud scheme and the individuals within and outside the organization who could be the perpetrators of each scheme. If the scheme is collusive, preventive controls should be augmented by detective controls, as collusion negates the control effectiveness of segregation of duties;

c. **Model Risk.** Model risk is considered a subset of operational risk, as model risk mostly affects the firm that creates and uses the model. Rebonato in [63] defines model risk as the risk of occurrence of a significant difference between the mark-to-model value of a complex and/or illiquid instrument, and the price at which the same instrument is revealed to have traded in the market.

1.2 Regulatory Capital and Basel Accords

Over the past 30 years, financial institutions, under the pressure and encouragement of the regulators, have developed advanced models in order to better face further financial crisis and correctly allocate the economic capital.
These models should help banks in managing and quantifying risk across geographical and product lines.
In particular, as a result of the economic crisis of 2007, the regulators have increased their interest regarding to the credit risk. The reason behind is that either an insufficient amount of capital reserves or an excessive exposure on specific asset classes can influence the solvency of a financial institution which, as result, can even led to an increase of the systematic risk.
Indeed, the strong connections among the financial institutions could bring the entire system to a unstable level. Often this is consequence is also known as "domino effect".
Those reasonable worries have triggered the authorities in accelerating the regulation processes that have taken form into a further regulatory developments. Nowadays, several indicators are used in order to help the banks in assessing the credit risk.
In 1983 the banking supervision authorities of the main industrialized countries (G7) agreed on rules for banking regulation, which should be incorpo-

rated into national regulation laws. In this sense a key role has been played by Basel Committee on Banking Supervision (BCBS) which was founded in 1974 as an international forum where members could cooperate on banking supervision matters. The BCBS objective, thanks to the introduction of a new set of rules known as accords, is to ensure the financial stability by improving the quality of banking supervision. Last 30 years has been characterized by 4 Basel accords:

1. **Basel I** [76]**.** The first accord has been signed in 1988 and introduced a a set of international banking regulations which established the minimum capital requirements of financial institutions with the aim of minimizing credit risk. The most famous rule introduced by Basel I is called the 8% rule. According to this rule, all banks that net international transactions have to reserve a minimum capital equal to the 8% of its RWA calculated for all balance sheet positions.

2. **Basel II** [4]**.** In 2004, following the implementation of the first international regulatory accord, Basel II introduced additional rules for minimum capital requirements, provided framework for regulatory review including how to face various types of risks (systematic risk, liquidity risk and legal risks), as well as set disclosure requirements for assessment of capital adequacy of banks and market discipline.
   A key change into the Basel II accord was the incorporation of credit risk of assets held by financial institutions for calculating regulatory capital ratios. Those ratios divide the eligible regulatory capital of a bank into three tiers. The higher the tier, the less subordinated securities a bank is allowed to include in it. Each tier must be of certain minimum percentage of the total regulatory capital and is used as a numerator in the calculation of regulatory capital ratios.
   Tier 1 capital is the most strict definition of regulatory capital that is subordinate to all other capital instruments, and includes shareholders' equity, disclosed reserves and retained earnings.
   Tier 2 is Tier 1 capital plus various other bank reserves, hybrid instruments, and medium-long term subordinated loans.
   Tier 3 consists of Tier 2 plus short-term subordinated loans.
   Basel II has redefined the definition of risk-weighted assets, which are used as a denominator in regulatory capital ratios, and are calculated by using the sum of assets that are multiplied by respective risk weights for each asset type. The idea behind the risk-weighted assets calculation is to punish banks which have risky positions, which significantly increases risk-weighted assets and lowers regulatory capital

ratios.
The regulatory capital ratios results to be more accurate in comparison to Basel I because the RWA calculation takes into account the credit rating of assets in determining risk weights. Greater will be the creditworthiness of the client, lower will be risk weight.
Besides, in the second international regulatory accord standardized approach (well known as SA approach) and internal model approach for market risk have been introduced. In this sense, Basel II played a relevant role in harmonizing the regulation going on among the different countries and alleviating anxiety over regulatory competitiveness and different national capital requirements for banks.

3. **Basel III** [8]**.** The third installment of the Basel Accords has been published as first version in late 2009, giving banks till 2015 for the implementation of all requirements at first glance.
This accord came out mostly in response to the deficiencies in financial regulation revealed by the financial crisis of 2007-2008, in order to increase the banking sector's ability to deal with financial stress, improve risk management, and strengthen the banks' transparency.
Basel III is intended to strengthen bank capital requirements by increasing bank liquidity and decreasing bank leverage:

   a. Basel III left the guidelines for risk-weighted assets largely unchanged from Basel II. Nevertheless, in comparison to Basel II, Basel III strengthened regulatory capital ratios, which are computed as a percent of risk-weighted assets.
   In particular, Basel III increased minimum Common Equity Tier 1 capital from 4% to 4.5%, and minimum Tier 1 capital from 4% to 6% where Common Equity Tier 1 capital includes equity instruments that have discretionary dividends and no maturity, while additional Tier 1 capital comprises securities that are subordinated to most subordinated debt, have no maturity, and their dividends can be cancelled at any time.
   The overall regulatory capital was left unchanged at 8%;

   b. The third installment of the Basel Accords introduced new requirements with respect to regulatory capital for large banks to cushion against cyclical changes on their balance sheets.
   In other words, banks have to reserve additional capital during period of credit expansion in order to be ready during the credit contraction where they are allowed to loosen the capital require-

ments.
Besides, banks are grouped according to their size, complexity and importance to the overall economy for facing the systemic risk. Indeed, big banks can have a large impact on the financial system causing a domino effect, so those banks are subject to higher capital requirements;

c. One of the objectives of Basel III is decreasing bank leverage and increasing liquidity requirements to safeguard against excessive borrowings and ensure that banks have sufficient liquidity during financial stress.
In particular, the leverage ratio, computed as Tier 1 capital divided by the total of on and off-balance assets less intangible assets, was capped at 3%.

4. **Basel IV.** Last Basel accord complements the initial phase of the Basel III reforms announced in 2010.
The 2017 reforms seek to restore credibility in the calculation of risk-weighted assets (RWAs) and improve the comparability of banks' capital ratios. Actually, Basel IV is a contested term, indeed most of the regulators believe that these changes are simply completing the Basel III reforms.
On the other hands, because this reform introduce a significant increase in capital requirements, it should be treated as a distinct round of reforms. Basel IV will be one of the major challenges for the financial markets in the next five years, because it will strongly impact the calculation of risk weighted assets and capital ratios of all banks and therefore will affect the banks' strategies.
The main items covered by this reform are:

a. Reforms of the Standardized Approach for credit risk. In particular, low risk mortgage loans will face a reduction of the risk weighted factors;

b. In the IRB-approach will be introduced a standardized floor, following the SA approach, equal to 72.5%, which will increase the capital requirements and will reduce the benefit for a bank to use the IRB approach to a maximum of 27.5%;

c. Shift to the Foundation internal ratings-based (F-IRB) approach for specific asset classes treated under IRB in the past;

d. Quantification of CVA risk;

e. Operational risk approaches;

f. A higher leverage ratio for Global Systemically Important Banks (G-SIBs), with the increase equal to 50% of the risk adjusted capital ratio.

1.3 Introduction to Credit Risk Metrics

Credit risk models, as underlined from Basel accords [4], [8], are not a simple extension of the market risk models for two main reasons:

1. **Data limitations.** Data limitations is a key impediment for implementing credit risk models. Most credit instruments are not marked to market, and the predictive nature of a credit risk model does not derive from a statistical projection of future prices based on a comprehensive record of historical prices.
   The scarcity of the data required to estimate credit risk models also stems from the infrequent nature of default events and the longer-term time horizons used in measuring credit risk;

2. **Model validation.** The validation of credit risk models is fundamentally more difficult than the backtesting of market risk models. Differently from market risk models where the time horizon is few days, credit risk models generally rely on a time frame of one year or more. The longer holding period, coupled with the higher confidence intervals used in credit risk models, presents problems to model-builders in assessing the accuracy of their models.
   In other words, for assessing the accuracy of the credit risk models are required several years of data, spread over multiple credit cycles.

Banks and regulators are highly investing in credit risk models for avoiding a further financial crisis. Indeed, credit risk models offer the opportunity to analyze in a timely manner, centralizing data on global exposures and have a deep look into marginal and absolute contributions to risk.
This means that credit risk models providing estimates of credit risk measure (such as unexpected loss) would be informative tool for risk management.
On the other hand, trying to reach more accurate estimates a lot of effort has been put in designing and building credit risk models. This can be seen as a starting point for improving systems and data collection infrastructure which will reflect in a better data quality and in more accurate output of the models.

Besides, better estimates will may contribute to a more transparent decision-making process and a more consistent basis for economic capital allocation. Hence, credit risk models may contribute to an improvement in a bank's overall ability to identify, measure and manage risk. The key role of the credit risk management in the banking sector can be easily clarified by an example.

Let us assume that a company which produces 3D printers is asking its house bank for a loan in the size of 100 million Euro. The bank has a credit department which has to decide if the credit request will be accepted or rejected.

Let us further assume that the analyst, who is taking care of the credit request, knows that the bank's chief credit officer has known the chief executive officer of the printer company for many years, and to make things even more complicated, the credit analyst knows from recent default studies that the 3D printers sector is under hard pressure and that the bank-internal rating of this particular company is slightly lower than investment grade (low credit quality).

Given the described situation, the analyst should reject the credit request and do not proceed with the deal.

An alternative would be to grant the loan to the customer and insure the potential loss through a credit risk management instruments. For reducing the lender's credit risk, the lender may perform a credit check on the prospective borrower, may require the borrower to take out appropriate insurance, such as mortgage insurance, or seek security over some assets of the borrower or a guarantee from a third party. The lender can also take out insurance against the risk or on-sell the debt to another company. In general, the higher the risk, the higher will be the interest rate that the debtor will be asked to pay on the debt. Credit risk mainly arises when borrowers are unable to pay due willingly or unwillingly.

A different example can be a bond issued by a corporation or sovereign government. In this case the credit risk might be viewed as part of market risk and one could argue to apply methods well known in market risk modeling also to credit risks.

There are multiple reasons which do not enable to apply the market risk mitigation techniques to credit risk. Firstly, this is related to the illiquidity of the credit-risky positions, for example bank loans, whose market prices are not readily determined. Second of all, default is a rare event, then historical default data is relatively sparse, compared to market prices and rates. Besides, the information needed for credit risk analysis is mainly contract-specific while prices and rates apply market-wide. Given these reasons,

credit risk techniques must differ from those specific to market risk analysis.

1.4 Risk Weighted Assets (RWAs)

Risk Weighed Assets (RWAs) are an important measure in the current credit risk environment.
Indeed, they represent an aggregated measure of different risk factors affecting the evaluation of financial products. In the RWA calculation several risk components are considered together to 'correct' the nominal value of financial assets. In this way, a proper measure of the extent to which the underlying risk is increasing or decreasing the accounting value of financial assets is generated.
This assessment attributes a high weight-coefficient to high-risk financial assets, and a low-weight coefficient to low-risk ones. A bank's assets typically include cash, securities and loans made to individuals, businesses, other banks, and governments.
Each asset have different risk characteristics to which is assigned a risk weighted factor (RWF) as indication of the risk taken by the bank in holding the asset and of how much capital banks should maintain to guard against unexpected losses.
In other words, banks need less capital to cover exposures to low risk assets and more capital for the riskier ones.
As the Basel Committee points out, RWAs play a very important role in the banking sector, helping banks monitoring their efforts in achieving capital adequacy goals.
As understandable from [1.2], the calculation of the amount of RWAs depends on which revision of the Basel Accords is considered.
In 1997 the Basel Committee on Banking Supervision allowed the banks to use so-called specific risk models, and the eligible instruments no longer fell under the 8%-rule. Hence, the standard risk weight factor under Basel I was 100%, which led to a regulatory capital of:

$$[risk\ weight] \times [solvability\ coefficiet] = 100\% \times 8\% = 8\%$$

Nevertheless, banks already introduced complicated internal models to handle the credit risk for their balance sheet positions with an emphasis on default risk.
This rule implied that the capital basis for banks was mainly driven by the

exposure of the loans to their customers. The main weakness of this capital accord was that it made no distinction between obligors with different creditworthiness, in other words the RWAs were not risk sensitive.
However, in 2004 the second Basel accords, which switched live in most banks worldwide on January 1st, 2007, did not allow the use of internal credit risk models for the calculation of regulatory capital.
Instead, they use a more complicated risk-weighting scheme for bank's credit risk positions. As underlined in [1.2], Basel III did not revolution the RWAs calculation, so for having a better understanding a good starting point is to show how calculation are intended according Basel II accords.
One of the biggest improvements of the second installment of the Basel Accords is a different parameterizations for different asset classes. For further details on the bucketing of the risk weighted factor is good to look at the guidelines given by the regulation in [4]. In the second Basel accord the RWAs assume much more sense, because they become risk sensitive weighting positions with reference to their credit risk.
In this context the most sophisticated approach which can be implemented by a bank is called internal ratings-based approach (IRB) and it is presented below according to the standard corporate loan guidelines:

$$RWA = 12.5 \times EAD \times LGD \times K(PD) \times M(PD, MAT)$$

where

$$K(PD) = N\left[\frac{N^{-1}[PD] + \sqrt{\phi(PD)}q_{99.9\%}(Y)}{\sqrt{1-\phi(PD)}}\right] - PD$$

and $\phi(PD)$ can be defined as:

$$\phi(PD) = 0.12 \times \frac{1-e^{-50\times PD}}{1-e^{-50}} + 0.24 \times \left(1 - \frac{1-e^{-50\times PD}}{1-e^{-50}}\right).$$

The meaning of the parameters in the formula is as follows:

- $M(PD, MAT)$ is an adjustment factor depending on the effective maturity (MAT) of the asset and its PD;
- $N[\cdot]$ is the standard normal distribution function and $N^{-1}[\cdot]$ is its inverse;
- The quantity $q_{99.9\%}(Y)$ is the 99.9%-quantile of a standard normal random variable Y;
- The quantity ϕ has the meaning of a correlation parameter;

- The formula for the correlation parameter ϕ is an interpolation between 12% and 24% quantifying the systematic risk (R^2).

A Clear strength of Basel II accords is that IRB-banks can use their internal ratings, $LGDs$ and $EADs$ as an input into the RWAs function.
This is a huge progress in Basel II compared to Basel I.
It basically means that regulatory and economic approaches exhibit high convergence for single-name risks.
On the other hand, it still presents weakness related to the diversification because it does not take in account either the concentration or diversification of the portfolio.
This cons becomes even more dramatic when related to the securitization because it completely ignore the effects and risks of such these so-called correlation products.
What comes next? Basel accords are in continuous development and the Basel IV accord published late 2017 promise to keep the banks busy at least for the next 4 years. Indeed, the implementation of the last Basel accord should be completed by the beginning of 2022 and should highly impact the RWAs as explained [1.2].

1.5 Expected Loss (EL)

Historical data shows that even good customers, who had always respected the repayment schedule, have a probability to unmeet their financial obligations.
Hence, the entire portfolio, and not only the high risk part, should be insured against potential loss.
This mechanism reflects the insurance structure common in other insurance sectors (car, health, etc..) and it will create a capital cushion for covering losses arising from defaulted loans charging an appropriate risk premium for every loan.
In other words, banks start to reserve money for fixing the roof before that the storm comes.

Definition 1 *The loss of any obligor can be formally defined by a loss variable:*

$$\tilde{L} = EAD \times LGD \times L \tag{1.1}$$

where EAD represents the exposure at default which could be lost in a specific time frame, LGD stands for the loss given default and describes

the fraction of the loan's exposure expected to be lost in case of default and with:

$$L = \mathbf{1}_D, \mathbb{P}(D) = PD$$

where D denotes the event that the obligor defaults in a certain period of time (often one year), and $\mathbb{P}(D)$ denotes the probability of D.
In other words the bank assigns a probability of default at each customer

The constituents of formula [1.1] are random variables which exist in a probability space (Ω,$\mathcal{F}$,$\mathbb{P}$), consisting of a sample space Ω, a σ-Algebra F, and a probability measure $\mathbb{P}$.
The elements of $\mathcal{F}$ are the measurable events of the model, and intuitively it makes sense to claim that the event of default should be measurable in a a specified time horizon.

Definition 2 *The expected loss (EL) of any customer can be defined as the expected value of [1.1]*

$$EL = \mathbb{E}[\tilde{L}]$$

which is called the expected loss of the underlying credit-risky asset.

Under the assumption that PD, EAD and LGD are independent in [1.1], the expected loss can be reformulated as follow:

$$EL = \mathbb{E}[EAD] \times \mathbb{E}[LGD] \times PD \tag{1.2}$$

However, if EAD and LGD are constant values, then [1.2] can be written as

$$EL = EAD \times LGD \times PD \tag{1.3}$$

where the expectation of the Bernoulli random variable Ł is its event probability.
If the three variables in [1.1] are independent the expectation of their product is the product of their expectation.
Nevertheless, in realistic situations EAD is not constant and it has to be modeled as a random variable due to uncertainties in payment profiles which are driving the exposure.
In [1.3], it is also assumed that all components (PD, EAD and LGD) are independent. Unfortunately, this assumption helps in the calculation but it is not verified in real life.
The lack of this assumption can be easily shown recalling the recent subprime mortgage crisis in the US. Due to a recession period the probability

of late payments increase; This situation generates an higher number of defaults. Most of the times the loans, mortgages in our example, are covered by collaterals, residential units, which banks can use for recovering their exposures. Moreover, in a recession scenario, there will be an over-supply of residential apartments put which will automatically lead to a price drop of the collateral.
In other words, banks will not be able to recover the expected amount by selling collateral.
This example shows how PD and LGD are positively related, indeed the number of defaults, strictly link to the increase of the probability of default given the recession scenario, cause an increase of the LGD due to the incapacity of banks in recovering the collateral value. Altman et al. in [2] give further details on this relations.
The EAD can be also considered dependent from PD because in times of financial distress firms tend to draw on their open credit lines and it cause an increase of the exposures when default rates are systematically going higher.

1.6 Probability of Default (PD)

The probabilities of default are a key topic for a credit risk analytics team which has to properly set the PD either for asking the correct risk prime to the client or booking the right amount of provisions.
Indeed, the correct assignment of default probabilities could be straightforward in some cases while in other situations seems almost impossible to derive a reasonable approach. It is important to underline that the default probabilities do not refer to a single point in time, but the entire term structure is needed:

$$(p_t)_t \geq 0$$

where t denotes time and for each point t in time p_t is the default probability of the considered asset or client with reference to the time interval $[0, t]$.
Note that in the literature PD term structures are often called credit curves.
For simplifying this introduction will focus on a fixed time horizon which is assumed to be one year.
As highlighted for calculating the probability of default in accurate way, it is relevant to estimate the creditworthiness of a client or asset. Financial institutions and external rating agencies measure the creditworthiness of those assigning a rating.
Nevertheless, rating estimation assume a different role between banks and

external rating agencies, the latter assign rating only for creating an ordinary scale of the credit quality. For example according to Moody's, one of the most important external rating agencies (other well-known rating scales are developed by the rating agencies Standard & Poor's and Fitch), a sovereign bond with 10 years maturity issued by the Netherlands is rated Aaa while the same kind of bond issued by Spain is rated Baa.
Moody's rating scale uses Aaa, Aa, A, Baa, Ba, B, Caa, etc. as values where the movement thought the alphabet letters denotes a decreasing of credit quality.
Each of the rating agencies has a finer rating scale in place to allow for a finer distinction of credit quality among obligors. From the figure [1] is clear that

Moody's	Standard & Poor's	Fitch	AM Best	Credit worthiness
Aaa	AAA	AAA	aaa	An obligor has EXTREMELY STRONG capacity to meet its financial commitments.
Aa1 Aa2 Aa3	AA+ AA AA-	AA+ AA AA-	aa+ aa aa-	An obligor has VERY STRONG capacity to meet its financial commitments. It differs from the highest rated obligors only in small degree.
A1 A2 A3	A+ A A-	A+ A A-	a+ a a-	An obligor has STRONG capacity to meet its financial commitments but is somewhat more susceptible to the adverse effects of changes in circumstances and economic conditions than obligors in higher-rated categories.
Baa1 Baa2 Baa3	BBB+ BBB BBB-	BBB+ BBB BBB-	bbb+ bbb bbb-	An obligor has ADEQUATE capacity to meet its financial commitments. However, adverse economic conditions or changing circumstances are more likely to lead to a weakened capacity of the obligor to meet its financial commitments.
Ba1 Ba2 Ba3	BB+ BB BB-	BB+ BB BB-	bb+ bb bb-	An obligor is LESS VULNERABLE in the near term than other lower-rated obligors. However, it faces major ongoing uncertainties and exposure to adverse business, financial, or economic conditions which could lead to the obligor's inadequate capacity to meet its financial commitments.
B1 B2 B3	B+ B B-	B+ B B-	b+ b b-	An obligor is MORE VULNERABLE than the obligors rated 'BB', but the obligor currently has the capacity to meet its financial commitments. Adverse business, financial, or economic conditions will likely impair the obligor's capacity or willingness to meet its financial commitments.
Caa	CCC	CCC	ccc	An obligor is CURRENTLY VULNERABLE, and is dependent upon favourable business, financial, and economic conditions to meet its financial commitments.
Ca	CC	CC	cc	An obligor is CURRENTLY HIGHLY-VULNERABLE.
	C	C	c	The obligor is CURRENTLY HIGHLY-VULNERABLE to nonpayment. May be used where a bankruptcy petition has been filed.
C	D	D	d	An obligor has failed to pay one or more of its financial obligations (rated or unrated) when it became due.
e, p	pr	Expected		Preliminary ratings may be assigned to obligations pending receipt of final documentation and legal opinions. The final rating may differ from the preliminary rating.
WR				Rating withdrawn for reasons including: debt maturity, calls, puts, conversions, etc., or business reasons (e.g. change in the size of a debt issue), or the issuer defaults.
unsolicited	unsolicited			This rating was initiated by the ratings agency and not requested by the issuer.
	SD	RD		This rating is assigned when the agency believes that the obligor has selectively defaulted on a specific issue or class of obligations but it will continue to meet its payment obligations on other issues or classes of obligations in a timely manner.
NR	NR	NR		No rating has been requested, or there is insufficient information on which to base a rating.

Figure 1: External Ratings

the sovereign bonds issued by the Netherlands are safer than the Spanish

ones. In other words, according to Moody's, even if a probability of default is not assigned to each rating, Spanish bonds have an higher probability of default of the Dutch ones.
Although, the higher risk of Spanish bonds will be compensated by an higher interest rate.
In the meanwhile banks associate to a rating a determinate probability of default. For example, if we assume a rating scale from 1 to 20, where 1 refers to a top client while 20 to a defaulted customer, all customers rated as 1 will have a probability of default equal to 0.01 while the ones in the bucket 20 will have probability of default equal to 1.
It is relevant to keep in mind that this figure is a over-semplification of the PD/rating scenarios. Indeed, banks associate different models to different exposure classes, in other words the probability of default assigned to a large corporate, rated as 3, will be different from a small enterprise with the same rating, because the different customers will fall in different rating models which will calculate the probability of default.
A rating system can be thought of as a discretization of PDs on an ordinal scale which is called the rating scale. Discretization of a continuous metric quantity like a PD to an ordinal scale makes life in large organizations easier although one could argue that discretization introduces unnecessary jumps in pricing grids.
The procedure of discretization of PDs, namely the assignment of a PD to every rating grade in the given rating scale is called a rating calibration; The latter is a relevant process which deserves few lines for showing, in a simplified way, how it works.
Given:

$$R : PD(R) \to [0, 1] \tag{1.4}$$

such that to every rating R a certain default probability PD(R) is assigned. Being in a probabilistic environment the calibration function is always in the interval [0, 1] and at each range of the probability of default is assigned a rating.
As described the rating allocation is a complex process which can be executed in multiple ways. A brief overview, following the schema given by Bluhm et al. in [10], about the main four rating systems will be shoved below:

1. **Causal Rating Systems.** As well described by Bluhm et al. in [10], Casual Rating Systems is considered a superior to all other approaches. Whenever possible, this should be the way to proceed. As the name indicates, causal rating systems rely in their mechanism on a causal

relationship between underlying credit risk drivers and the default event of an asset or borrower. To mention an example as explained by Bluhm et al. in [9], ratings assigned to tranches in collateralized debt obligations (CDO) typically are of causal type because the CDO model derives scenarios where the considered tranche is hit by a loss as well as the loss severity of the tranche as a direct consequence of "turbulences" in the underlying reference portfolio of credit-risky instruments.
The model-derived hitting probability, for instance, can then be mapped onto a rating scale such that, for instance, a tranche with low hitting probability might have a letter combination like AAA or AA whereas a tranche with a low capital cushion below might get a rating letter combination of B or even in the C-range.
Why causal rating models are considered the best way to think about ratings?
The reason is that a causal model approach forces the modeler to extensively analyze, understand and model the "true" mechanism of default. This is under all circumstances the best a modeler can do.
For instance, a CDO model requires a fully-fledged model for both the cash flow structure of the CDO as well as the credit risk of the underlying reference portfolio. Causal models force the modeling team to really understand how defaults can happen and how losses will accumulate under certain circumstances.
The most representative of this type of rating systems is the concept of Expected Default Frequencies (EDF) from Moody's KMV [21].

2. **Balance Sheet Scorings.** In some cases the above mentioned causal approach is rather difficult or even impossible to apply because the default mechanism cannot be modelled. In such cases one can switch to scoring systems which are a good choice and well-established in rating units in banks across the globe.
For instance, whereas for stock exchange-listed corporate clients a causal modeling of PDs is market standard as mentioned before, it is hardly thinkable to follow a causal approach for private corporates. Moreover, there are many companies which do not have a so-called external rating, which is a rating assigned by the afore-mentioned rating agencies.
In such cases, a balance sheet scoring model is the usual approach to assign a bank-internal rating to such companies.
In general, banks always should base the decision about creditwor-

thiness on their bank-internal rating systems. As a main reason one could argue that banks know their customers best.
Moreover, it is well known that external ratings do not react quickly enough to changes in the economic health of a company. Banks should be able to do it better based on their long-term relationship with their customers.This is typically done by the credit analysts of the bank based on the rating tools developed by the rating quant team.
For rating assignment the credit analysts consider various different quantitative and qualitative drivers of the considered firm's economic future like, for instance:

(a) Future earnings and cashflows;

(b) Debt, short and long-term liabilities, and financial obligations;

(c) Capital structure (i.e., leverage);

(d) Liquidity of the firm's assets;

(e) Situation (i.e., political, social, etc.) of the firm's home;

(f) Situation of the market (i.e., industry), in which the company has its main activities;

(g) Management quality, company structure, etc.

From this list, it can be understood that rating drivers can be quantitative as well as qualitative.
To mention another important example, succession planning can be important for smaller firms but can not be captured as a solid quantitative figure like, for instance, a debt-equity relation.
It is best practice in banking that ratings as an outcome of a statistical tool are always re-evaluated by the credit analyst who makes the credit decision which leads to 'approved' or 'rejected'. Credit analysts typically have, in line with their credit decision competence, the right to overrule or override the calculated rating. In most of the cases this will be an override to a better or worse rating grade by not more than one or two notches.
The overruling quote which measures the relation of overruled ratings compared to overall assigned ratings is a good measure of the acceptance of a rating system by the practitioners in the credit unit, namely, the credit analysts who distinguish themselves from the rating quants who developed the rating system. More information related to the internal rating system are showed by Fritz et al. in [37].
Overruling competence is crucial because especially for smaller firms

one can expect that certain aspects driving the ability to pay of the client might not be captured by a standardized statistical tool.
The afore-mentioned quantitative drivers of the rating are taken from the balance sheet and annual report of the borrowing company. These sources of information are important for the lending credit institute because it is pretty much all one can get if a company is not listed at an exchange and has no public debt outstanding. The afore-mentioned rating drivers are then grouped and set in relation to form a list of balance sheet ratios which are mapped into so-called scores as a metric-scale measure of default remoteness. The total score of a client is then based on a weighted sum of ratio transformation functions, often involving a lot of regression analysis in the development process. The score of a client is then calibrated to a PD based on the history of default frequencies:

$$PD_{client} = \frac{1}{1 + exp(-SCORE_{client})} \tag{1.5}$$

where $SCORE_{client}$ represents the final score based on the afore-mentioned sum of transformed ratios.
The PD_{client} showed in [1.5] is representative of the class of so-called logit calibration functions which is a common transformation approach to get PDs out of scores in balance sheet scorings.
An industry example for an off-the-shelf model to obtain ratings for private companies is the RiskCalc model by Moody's KMV showed in [29].

3. **Private Client Scorings.** In the same way as one can build scoring systems for private companies one can derive scoring systems for private individuals, for instance, for clients which borrow money in the context of a residential mortgage.
The basic mechanism is exactly the same but the rating score drivers are different.
For instance, personal wealth, income situation, family context, etc. are typical drivers for a private client scoring.
Moreover, practitioners know that the main drivers for default in a residential mortgage lending context and, more general, in any lending to private individuals are unemployment, divorce, and poor health.

4. **Expert Rating Systems.** There are portfolios where over many years hardly any default occurred.

For example, municipalities in Switzerland can be grouped into a portfolio where almost no defaults occurred.
In such situations it is difficult to work with balance sheet scorings because the number of defaults in the portfolio is too low for deriving statistically sound conclusions.
This kind of portfolios where a default event barely occur are defined as low default portfolios and how to approach to those is well explained by Wilde and Jackson in [75].
A common approach then is to overcome the problem of missing defaults by involving groups of experts in the considered segment in the bank to assign manual ratings to test cases.
These kind of portfolios where it is difficult to use time series analysis for default estimates are under the radar of the regulators and financial supervisors. Indeed, i.e. Basel IV accords are imposing to the banks to move portfolio composed by Institutions under the F-IRB approach which means that banks cannot use own LGD's estimates, but a pre-defined value fixed at 45%.

It is important to note that the rating type categories are not fully disjoint. In many cases a rating system has a main flavor but it is combined with technology from some other rating model types.
For instance, a rating model could be causal in principal but also use elements from scoring theory and regression [6].

1.7 Exposure At Default (EAD)

As defined in (1.1) the EL is the product of three variables and Exposure at Default (EAD) is one of those.
Hence, the EAD quantifies the amount that has been borrowed and not repaid yet at which the bank will be exposed in case of default. A company either for investing or keeping the activity going on could decide to finance itself with a bank loan, once the loan is approved, a credit line will be opened for that company.
This credit line represents the maximum amount which can be drawn by the company. At this stage, it is important to introduce the concepts of sub-limits and outstandings for having a better clue about the credit structure. Let us assume that a company has a credit line of EUR $10 Mio$, also called cross facility or cross limit which can be divided in sub-limits/sub-facilities. Keeping the structure simple, we make the assumption that the cross credit

limit of EUR $10 Mio$ has been divided in two sub-limits, then the credit situation could be rewritten in the following way:

1. **Sub-Limit.** The borrower can draw EUR $6 Mio$ as cash;
2. **Sub-Limit.** The rest of the credit line (EUR $4 Mio$) for so-called contingent liabilities, i.e., guarantees or comparable credit constructs but not for cash.

Once the company receives all approvals from the bank for opening the credit line, it can start to draw from it.
The drawn amount is called outstanding, while the remaining part of the credit lines represents the commitments that the bank has towards the borrower.
Let us assume that the borrower draws EUR $5 Mio$ from the first sub-limit then this amount will represent the outstandings while the remaining part (EUR $1 Mio$) is seen as a commitments for the bank because the borrower can decide to draw it at anytime.
In other words, the outstandings refer to the portion of the overall client exposure that the obligor is already using. There is not randomness involved, drawn is drawn, and if the obligor defaults then the outstandings are subject to recovery and in a worst case situation could potentially be total lost.
On the other hand, to evaluate at priori the exposure of the bank on the remaining EUR 5Mio, the commitments, is tricky. The most accurate way, and most probably the only one, for considering the exposure arising from the open part of the credit line is a random variable. In conclusion, the two parts of the open line address different random effects.
The EUR 1Mio which can be drawn as cash are driven by the likelihood that the borrower draws on them as well as by the fraction quantifying how much of the 1Mio he draws.
Bringing this concept to a more formal environment, it could be explained by the following expression:

$$EAD_{cash} = 1_D \times X \times [1 Mio](EUR) \tag{1.6}$$

for the random exposure adding to current outstandings.
Here, D describes the event (in the σ-field $\mathcal{F}$) that the obligor draws on the open cash credit line and X is a random variable defined on the underlying probability space (Ω, $\mathcal{F}$, $\mathbb{P}$) with X (ω) $\in$ [0, 1] for each $\Omega \in \omega$ quantifying the random fraction describing how much of the open EUR $1 Mio$ line is drawn.

The remaining EUR 4Mio which can be used for contingent liabilities are also subject to various random effects.
First of all, there are again one or more indicator variables reflecting the optionality of usage of free parts of the second sub-limit. Second, there is randomness in the fact that contingent liabilities not necessarily lead to cash exposure. A guarantee has no real exposure as of today but might converge into exposure in the future. Such random effects are typically treated by so-called conversion factors.
One common exposure parameter is the so-called draw-down factor (DDF). In our example it could be the case that the bank is able to say that the given type of obligor tends to draw on the free part of the credit line (EUR 1Mio) in 80% of the cases and on average uses 60% of the available cash.
In other words, based on historic experience the bank obtains parameters in [1.6] like

$$\mathbb{P}(D) = 80\% \; and \; \mathbb{E}[X] = 60\%$$

Assuming that $1_D \perp X$, this leads to an expected cash exposure for the unused part of the cash credit line of:

$$\mathbb{E}[EAD_{cash}] = \mathbb{P}(D) \times \mathbb{E}[X] \times [1Mio](EUR) = 48\% \times [1Mio](EUR)$$

The 48% would then be used as the DDF for this particular situation. Note that the DDF is one particular common example for conversion factors.
For the contingent liability part of the credit line we assume again the existence of a rich database which allows for the calibration of a DDF of, say, 40% for the contingent liability part and a so-called cash equivalent exposure factor (CEEF) of 80% which is another conversion factor quantifying the conversion of the specific contingent liability, say, a guarantee, into a cash exposure.
Altogether, we obtain, under the independence assumption, the following representation for the EAD:

$$\mathbb{E}[EAD] = [5Mio] + 48\% \times [1Mio] + 32\% \times [4Mio](EUR)$$

which end up in:

$$[5Mio + 0.48Mio + 1.28Mio](EUR) = [6.76Mio](EUR)$$

where

$$32\% = 40\% \times 80\%.$$

So altogether our (expected) EAD is between the already utilized $5Mio$ and the overall committed $10Mio$ but higher than the committed cash line

of $6Mio$.
Our example provides an idea on how complicated EAD calculations can be, in practice, in the real life, it is even more complex.
For example, commitments of banks to clients often include various so-called covenants, which are embedded options which, for example, may force an obligor in times of financial distress to provide more collateral or to renegotiate the terms of the loan.
A problem is that often the obligor has some informational advantage in that the bank recognizes financial distress of its borrowers with some delay. In case of covenants allowing the bank to close committed lines triggered by some early default indication. This is a matter of timing, the bank has to pick such indications early enough to react before the customer has drawn on his committed lines.
Bankers here often speak of a race to default which addresses the problem that distressed clients tend to exhaust their lines just before they default as much as possible.
The Basel Committee on Banking Supervision in [23] provides conversion factors for banks who are unable or not allowed by their regulator to calibrate their own internal conversion factors like DDFs and CEEFs.

1.8 Loss Given Default (LGD)

Differently from EAD, the LGD is commonly expressed in percentage.
Hence, the loss which occurs in case of default will be denoted as $\$LGD$.
Let us assume that a client has m credit products with the bank and pledged n collateral securities to the bank which can in case of default be used for recovery purposes in order to mitigate the realized loss arising from the client's default.
As described in [1.7], once the bank open a credit line it gets an exposure; most probably the cross facility is split in several sub facilities which refers to m credit products.
In other words we can say that the bank get $EAD_1, \cdots, EAD_m$ for each of the m products.
In some cases the bank can ask to have some guarantees from the customer which can be sold in case of default.
The easiest example is the mortgage where the house represents a guarantees itself. Leaving the residential mortgages field, and extending our example to a company we can expect a recovery which proceeds from the n collateral securities.

The recovery proceeds can be denoted $\$REC_1, \cdots, \$REC_n$.
This is a m to n situation, indeed the several exposures can be covered from one single collateral which creates tricky to build an interdependence between products and collateral, especially in cases where we have to deal with dedicated collateral which can be used for certain purposes and only under certain circumstances.
Under a simplistic approach, it can be assumed that only "good cash" (recovery proceeds) and "bad cash" (loss exposure) can be gathered. So, we have two separate buckets which we then compare to obtain our net balance with the defaulted client:

$$\$LGD = max(0, (EAD_1 + \cdots + EAD_m) - (\$REC_1 + \cdots + \$REC_n)) \quad (1.7)$$

which leads to a percentage LGD of

$$LGD = \frac{\$LGD}{EAD_1 + \cdots + EAD_m} \quad (1.8)$$

The m to n relation between recoveries and exposures is not the only complexity that we have to highlight in the LGD calculation.
Indeed, the $\$REC_i$ quantities are not easy to derive given the complexity and granularity of the data required.
Besides, the time value of money is another known issues, due to the kind of guarantee the recovery proceeds coming later in time respect to the default, so it could not reflect the time value of money and an appropriate discount rate has to be estimated.
In other words, LGD calibration is a long story and far from being trivial. In this sense, the current regulatory framework is taking into account the difficulties which banks are facing in estimating the LGD and it is forcing them to treat some asset classes under the F-IRB instead of A-IRB approach.

1.9 Unexpected Loss

In the previous sections, most of the Credit Risk metrics such as RWA in [1.4], EL in [1.5], PD in [1.6], EAD in [1.7], LGD in [1.8] have been introduced.
Nevertheless, as well explained by Bluhm et al. in [10], it is possible to look at the the credit risk from a different perspective.
Indeed, Credit Risk can be generalized with the following equation:

$$CreditRisk = \max(ActualLoss - ExpectedLoss, 0)$$

where the actual loss is the observed financial loss. Hence, credit risk is the risk that the actual loss is larger than the expected loss.
Expected loss (EL) is an estimate and it has been previously introduced [1.5] as an insurance or loss reserve in order to cover losses the bank expects from historical default experience.
But a focus on expected losses is not enough, because actual loss could be considerably larger the the expected loss.
In fact, the bank should in addition to the expected loss also make sure that they have a good understanding on how much money would be necessary for covering unexpected losses where 'unexpected' stands for losses exceeding the historic average observed in the past.
As a measure for the magnitude of the deviation of losses from the EL, the standard deviation of the loss variable $\tilde{L}$ as in [1.1] could be a good starting point.
Hence, the unexpected loss unexpected loss of the underlying loan or asset is defined as:

$$UL = \sqrt{\mathbb{V}[\tilde{L}]} = \sqrt{\mathbb{V}[EAD \times LGD \times L]} \tag{1.9}$$

Following the formula shown in [1.9] and under the assumption that EAD is deterministic and that LGD and the default event D are independent, the unexpected loss can be re-written as:

$$UL = EAD \times \sqrt{\mathbb{V}[LGD] \times PD + \mathbb{E}[LGD]^2 \times PD(1 - PD)} \tag{1.10}$$

Nevertheless, banks struggle with calculating the UL of huge portfolios which contain many different products with different risk characteristics.
Hence, starting from a family of m loans the $\tilde{L}_i$ of the i asset with i=1, 2, $\cdots$,m is given by

$$\tilde{L}_i = EAD_i \times LGD_i \times L_i \tag{1.11}$$

with

$$L_i = 1_{D_i} \; and \; \mathbb{P}(D_i) = PD_i$$

The loss of the whole portfolio $\tilde{L}_{PF}$, containing homogeneous assets, can be defined as a collection of $\tilde{L}_i$ loss variables stated in [1.11].
Hence, the portfolio loss is then defined as the following random variable:

$$\tilde{L}_{PF} = \sum_{i=1}^{m} \tilde{L}_i = \sum_{i=1}^{m} EAD_i \times LGD_i \times L_i \tag{1.12}$$

Given a portfolio of m loss variables defined in [1.11], the expected and unexpected loss of the entire portfolio can be written as:

$$EL_{PF} = \mathbb{E}[\tilde{L}_{PF}] \; and \; UL_{PF} = \sqrt{\mathbb{V}[\tilde{L}_{PF}]} \tag{1.13}$$

Nevertheless, the loss variables within the UL_{PF} are correlated and this correlations between single assets play a fundamental role.
Given a portfolio of m loss variables as in [1.11] with deterministic EAD's, the portfolio UL is given by

$$UL_{PF} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} EAD_i \times EAD_j \times Cov[LGD_i \times L_i, LGD_j \times L_j]} \quad (1.14)$$

If we assume the LGDs are also deterministic and defining $Cov[L_i, L_j]$:

$$Cov[L_i, L_j] = \sqrt{\mathbb{V}[L_i]\mathbb{V}[L_j]}Corr[L_i, L_j]$$

and

$$\mathbb{V}[L_i] = PD_i(1 - PD_i) \; for \; each \; i = 1, \ldots, m$$

The previous formula [1.14] can be rewritten as:

$${UL^2}_{PF} = \sum_{i=1}^{m}\sum_{j=1}^{m} EAD_i \times EAD_j \times LGD_i \times LGD_j \times \sqrt{PD_i(1 - PD_i)PD_j(1 - PD_j)}\rho_{ij}$$

where the default correlation between the i and j counterparties is denoted by $\rho_{ij} = Corr[L_i, L_j]$.
It is relevant to understand how correlation can be interpreted in the 'real world'.
Let us consider a portfolio consisting of two loans with $LGD = 100\%$ and $EAD = 1$.
We then only deal with L_i for $i = 1, 2$ and we set $\rho = Corr[L1, L2]$ and p_i=PD_i.
Then, ${UL^2}_{PF}$ is given by:

$${UL^2}_{PF} = p_1(1 - p_1) + p_2(1 - p_2) + 2\rho p_1(1 - p_1)p_2(1 - p_2) \quad (1.15)$$

Let us focus on the extreme values which can be assumed by the default correlation ρ:

1. $\rho = 0$. In this case, the third term in [1.15] disappear.
   This kind of situation means that the two loans composing the portfolio are not linearly related. Financially speaking, $\rho = 0$ stands for the achievement of the optimal diversification.
   The concept of diversification is well known and easy to explain.
   If we consider to give two loans to two different companies, in two different sectors which are in two different countries where the companies, the

countries and the sectors are not related to each other, if one of the two company goes in default, there is no reason to suppose that the other company is going to default as well.
Obviously, nowadays, to have two loans perfectly unrelated seems to be only a theoretical situation.
Using the UL_{PF} in [1.15] as estimate of the portfolio risk, $\rho = 0$ represents a situation which minimize risk of joint defaults.

2. $\rho = 1$. This is a case of perfect correlation where the default of one obligor makes the other obligor defaulting almost surely.
$\rho = 1$ essentially meaning that our portfolio contains the risk of only one obligor but with double intensity the so-called concentration risk.
Which formally means that [1.15] can be re-written as follow:

$$UL_{PF} = 2\sqrt{p(1-p)}$$

Cases less extreme are represented from $0 < \rho < 1$ where the two counterparties borrowing money are positively correlated and the default of one counterparty will increase the probability that the other counterparty will default as well.

3. $\rho = -1$. This is the inverse situation of the case $\rho = 1$.
In this case, it makes much more sense to talk about the market value of the assets composing the portfolio.
In this context, a marginal increase in market value of one of the loans immediately would imply a decrease in market value of the other loan.
From [1.15], it follows that UL_{PF} completely vanishes ($UL_{PF} = 0$).

Chapter 2

Machine Learning Techniques

As seen in the previous chapter [1.9], either for regulatory or business purposes, the credit risk management is mainly based on credit risk models. These models help in judging the creditworthiness of a client, in measuring the exposures which the bank has to face in case of the client's default and the assets that the bank has to reserve for tackling a default event.
By definition a model is an approximation of the reality; in a financial environment there is a recurring question related to the accuracy and reliability of these models.
How far are they from the reality?
However, the output of these models do not have dependencies only from the model structure. Indeed, the models' outcome is a result of several calculations, under some assumptions, based on several input.
In other words, a model is able to "reproduce" the reality if it can rely on proper foundations, where for the latter we intend the input data.
The credit risk model accuracy, as all models, does not depend only on the effectiveness, parametrization and complexity of the model but from the data which are inputed. In a pragmatic language, this situation would be summarized as "Garbage IN is equal to Garbage OUT".
In the financial environment, a proper estimation of the risks is crucial for a safe activity of the bank and for the financial system itself, given the strong interdependencies. Hence, one of the common goals for the banks which want to run an healthy business is to increase the knowledge of the own data and retrieve some insights out of those, which can be transformed in added value.

It is important to highlight that over the past 5 years, the amount of data is exponentially increased, either due to the digitalization process or to the increasing regulatory requests. The latter, for some of the main European players, could be exemplified with an huge growth of the delivered data points to the regulators which raised from 2000 in 2012 to 200000 in 2017 (info to be double checked).
Most of the times, this hunger of data have brought banks in collecting more and more information without building a proper infrastructure which enables to check the authenticity of the data.
In this context, where the data quality is not fully ensured during the collection of those, it becomes relevant to use a model for pointing out the data issues.

Definition 3 *The knowledge acquisition process within data is well known as Knowledge Discovery in Databases (KDD).*
This is a framework which has been defined in [32] as:

The non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

This is an interactive and iterative process involving several steps: selection, processing, transformation, Data Mining, interpretation/evaluation.
As well explained in [49], KDD tasks can be classified into four general categories:

a. dependency detection;

b. class identification;

c. class description;

d. exception/outlier detection.

Most of the works related to KDD (i.e. [28], [40], [55], [54], [77]) have focused the attention on the first three categories, which consists in finding "large patterns" in data, or in other words, in identifying characteristics of the data that are representative of a significant portion of the dataset.
The outliers identification is part of the KDD, the fourth category, and it will be the focal point of this thesis. This category focus on the data points that are not falling into the "large patterns" or outliers. In general, these are often ignored or discarded as noise.

2.1 An Introduction to the Anomalies Detection

Anomalies are data patterns that have different data characteristics from 'normal' instances.
The techniques used to identify unusual patterns that do not conform to expected behavior are known as anomaly detection techniques.
In other words, these techniques highlight small patters which deviate from the large ones. The identified data points through these techniques are called outliers.
Nevertheless, these small patters contain useful information and cannot be simply discarded.
As pointed out by Knorr and Ng in [49]: "One person's noise is another person's signal".
In other words, outliers are part of the data collection and they can provide useful insights.
Mining for outliers has several applications in different fields: telecom, credit card fraud, loan approval, pharmaceutical research, weather prediction, financial applications, marketing, customer segmentation, data cleaning, etc.
Before to further proceed is good to mention some of the statistical definitions of outliers found in literature:

a. 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 [62];

b. An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism [44];

c. An outlier is a data point that deviates by a certain standard deviation from the mean [30];

d. Observations are called outliers when their number is significantly smaller than the proportion of nominal cases, typically lower than 5% [25].

According to the different anomaly detection techniques an outlier definition can be generated.
As pointed out by Domingues et al. in [25], outlier detection is a notoriously hard task which become even trickier when, for example, an input dataset contaminated by outliers may degrade the final model if the training algorithm lacks robustness.
From a broaden point of view, the anomalies, which are the detected through data anomalies techniques, as explained in [20] can be categorized as:

a. **Point anomalies.** A single instance of data is anomalous if it's too far off from the rest. Business use case: Detecting credit card fraud based on "amount spent."

b. **Contextual anomalies** The abnormality is context specific. This type of anomaly is common in time-series data. Business use case: Spending 100 on food every day during the holiday season is normal, but may be odd otherwise.

c. **Collective anomalies.** A set of data instances collectively helps in detecting anomalies. Business use case: Someone is trying to copy data form a remote machine to a local host unexpectedly, an anomaly that would be flagged as a potential cyber attack.

The simplest approach for identifying irregularities in data is to flag the data points that deviate from common statistical properties of a distribution, including mean, median, mode, and quantiles.
Starting from one of the definitions mentioned in [2.1] an outlier can be considered a data point that deviates by a certain standard deviation from the mean.
Traversing mean over time-series data isn't exactly trivial, as it's not static. You would need a rolling window to compute the average across the data points.
Technically, this is called a rolling average or a moving average, and it's intended to smooth short-term fluctuations and highlight long-term ones.
Mathematically, an n-period simple moving average is defined as a low pass filter which is a less sophisticated version of the Kalman filter. The main short-comes of this approach are:

1. The data contains noise which might be similar to abnormal behavior, because the boundary between normal and abnormal behavior is not well defined;

2. The definition of abnormal or normal may frequently change, as malicious adversaries constantly adapt themselves. Therefore, the threshold based on moving average may not always apply;

3. The pattern is based on seasonality. This involves more sophisticated methods, such as decomposing the data into multiple trends in order to identify the change in seasonality.

However, the ability to detect anomalies has significant relevance, and anomalies often provides critical and actionable information in various application

domains.
At this stage is important to understand how the anomaly detection techniques can be applied in different situations of the banking environment:

a. **Improving the data collection process within banks**. According to the kind of client, the data collection process can have a different upload schedule (daily, weekly, monthly, etc..). Most of the big banks have a data collection process which locally gather data and deliver them to the central system.
Let us say that an Italian bank called A_{it} opens a credit facility to the brewery B, which has residence in the Netherlands, through the Dutch branch of the bank called A_{nl}.
All customer information will be gathered by A_{nl} and sent to the headquarter A_{it} which has the central data system where data of all local branches are collected.
This data collection can include some noise, in other words, some data can be wrongly delivered to the central system.
Once these anomalies have been identified, they cannot be simply deleted because they contain useful information for some specific clients, but a proper process for data correction has to be put in place.
Why noise cannot be simply cleaned?
In this context, a trivial example is provided by monitoring reports on RWAs movements overtime which banks have to provide to the regulator. Looking at the monthly time series of the risk weighted assets (RWAs) for the brewery B, it can be observed that the level of the RWAs for the past 10months is around 1Bio and in the current month is 10Bio.
This shift can suggest a data quality issue, but it can, partially, depends on business reason (a new credit facility, acquisition by a bigger brewery, change in the creditworthiness, etc..).
Hence, it is fundamental to identify which portion of this shift is due to anomalies into the data collection process and which part is driven by change in the business;
This record regarding the brewery is part of the data collection and it has to be delivered to the regulator as in the past months, it cannot be simply cleaned up.

b. **Credit card fraud detection.** A relevant anomalies detection application is related to the credit card frauds which mostly raise from the illegal use of lost or stolen credit cards.
Since credit card companies assume liability for unauthorized expenses on lost or stolen cards, identifying and preventing illegal use become cru-

cial for running a profitable and safe business.
In this context the contraposition between large patterns, credit card usage prior to being stolen, and small patterns, the new usage allows to detect the credit card fraud.

Most of the studies in anomalies detection consider being an outlier as a binary property.
Nevertheless, for many applications, the situation is more complex and it becomes more meaningful to assign to each object a degree of being an outlier. To tackle this short coming the local outlier technique (LOF) has been introduced [18] which assign for each object in the dataset a degree of outlierness.
In other words this algorithm also quantifies how outlying an object is. The outlier factor is local in the sense that only a restricted neighborhood of each object is taken into account which differs from clustering which consider the global picture.
Besides, these techniques are computationally inefficient so classification algorithms are taken as a valid alternative [12].
However, clustering algorithms play a role in the outliers detection where outliers are defined as objects not located in clusters of a dataset, usually called noise [45].
The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its clustering parameters. In other words they focus on the large patterns and only a few approaches are directly concerned with outlier detection. Furthermore, based on these clustering algorithms, the property of being an outlier is again binary.
Different and quite new approach to the anomaly detection is provided by by Liu et al. in [51] and [52]. According to this technique the anomalies are susceptible to a mechanism called isolation which enables isolation trees to exploit subsampling to an extent that is not feasible in existing methods.
In conclusion we can say that anomalies detection techniques can be classified as:

a. Statistical Methods;

b. Density-Based Anomaly Detection;

c. Clustering-Based Anomaly Detection;

d. Classification-Based Anomaly Detection[1];

[1] Among these models, we find the Support Machine-Based Anomaly detection which

e. New Anomaly Detection techniques such as Isolation Forest;

Given the huge number of techniques per each class of anomaly detection approach, we will focus only on some those.

2.2 Density-Based Anomaly Detection

In literature anomaly detection has been extensively studied [3] and [44], they provide a good wrap up of the all statistical methods available.
Nevertheless, these techniques are not the focal point of this thesis.
Following [3], it can be deducted that these techniques require a-priori knowledge based on:

a. Data distribution;

b. Whether or not the distribution parameters (e.g., mean and variance) are known;

c. Number of expected outliers;

d. Types of expected outliers (e.g., upper or lower outliers in an ordered sample) .

which the user may not have.
In density-based anomaly detection, as explained by Tan in [69] anomalies are defined data points that belong to regions of low density and measured as:

a. The reciprocal of the average distance to the k-nearest neighbours;

b. The count of points within a given fixed radius.

Hence in these kind of data anomaly techniques a standard distribution (e.g. Normal, Poisson, etc.) is used to fit the data, the outliers are defined based on the probability distribution.
As showed in [3] over one hundred tests of this category, called discordancy tests, have been developed for different scenarios.
Beyond the lack of the knowledge of the data distribution, a main problem is linked to the number of attributes. Indeed, most of these tests are univariate which make them unsuitable for multidimensional datasets and to

in the middle of 1990's gave an applicative boost to the Statistical learning theory which was mostly used for theoretical analysis. A good overview of the Statistical Learning theory is given by Vapnik and Chervonenkis in [74], Vapnik in [72] and [73].

the assumptions on the data distribution.
In order to overcome the above-mentioned problems described as distribution fitting and univariate constrain, some computational methods have been developed, which can be best described as depth-based.
These methods are well presented in [61], [64] and based on some definition of depth, where data objects are organized in layers of the data space, with the expectation that shallow layers are more likely to contain outlying data objects than the deep layers.
However, in practice, the computation of k-dimensional layers relies on the computation of k-dimensional convex hulls where the lower bound complexity of those is Ω of order $N^{\lfloor \frac{k}{2} \rfloor}$, which means that depth-based methods are not expected to be practical for more than 4 dimensions for large datasets.
In fact, as showed in [64], the existing depth-based methods only give acceptable performance for $k \leq 2$.
More details on the depth definition are provided, in the univariate case, by Tukey in [71] and, in the multivariate case, by Donoho and Gasko in [26].
This notion of outliers is different either from the above-mentioned statistical definitions [2.1] of outliers or from the notion of outliers considered in [49] which associates a notion of distance and similarity measure between objects:

Definition 4 *An object O in a dataset T is a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O.*

The term $DB(p, D)$-outlier stands for a Distance-Based outlier. This definition is aligned with the one provided by Hawkins in [44].
Besides, it tackles some of the issues faced by the statistical approaches in [3] given that the assumption on the distribution is not needed and it overcomes the computational problems faced by the depth-based methods which are constrained by a small values of k dimensional layers.
Indeed, depth-based methods rely on the computation of layers in the data space while DB-outliers do not take in consideration the data space and rely on the computation values based on a metric distance function.
In distance based anomaly detection, anomalies are defined to be data points which are distant from all other points.
As showed by Tan in [69], two common ways to define distance-based anomaly score are:

a. The distance to k-th nearest neighbour;

b. The average distance to k-nearest neighbours.

In [49], the adopted metric distance function is (weighted) Euclidean.

Definition 5 *Let N be the number of objects in dataset T, and let F be the underlying distance function that gives the distance between any pair of objects in T. For an object O, the D-neighborhood of O contains the set of objects $Q \in T$ that are within distance D of O:*

$$Q \in T | F(O, Q) \leq D \tag{2.1}$$

The fraction p is the minimum fraction of objects in T that must be outside the D-neighborhood of an outlier. For simplicity of discussion, let M be the maximum number of objects within the D-neighborhood of an outlier, i.e., $M=N(1\text{-}P)$.

In other words given p and D, the problem of finding all $DB(p, D)$-outliers can be solved by finding a nearest neighbor or range query centered at each object O. More specifically, based on a standard multidimensional indexing structure, we execute a range search with radius D for each object 0.
As soon as (M+1) neighbors are found in the D-neighborhood, the search stops, and O is declared a non-outlier; otherwise, O is an outlier.
The lower bound complexity for a range search is Ω of order $N^{[\frac{1-1}{k}]}$, where k is the number of dimensions or attributes and N is the number of data objects.
As k increases, a range search quickly reduces the order to N, giving at best a constant time improvement reflecting sequential search. This is defined as index based procedure for finding all $DB(p, D)$-outliers and it has a worst case complexity of order (kN^2).
This index-based algorithm scales much better with dimensionality compared to the depth-based one, which have a lower bound complexity of Ω of order $N^{[\frac{k}{2}]}$ that means that the DB-outliers is applicable and computationally feasible for datasets that have many attributes, i.e., $k \geq 5$.
As described in [64], this is a significative improvement with respect to the existing methods which can only deal with $k \leq 2$.
Nevertheless, the index-based procedure for finding all $DB(p, D)$-outliers only considers search time which is a strong assumption that the right index exists.
In other words, the index-based algorithms are uncompetitive because it requires to build an index for finding all $DB(p, D)$-outliers. A valid alternative is provided by a block-oriented, nested-loop (NL) design presented by Knorr and Ng in [49].
The algorithm divides the dataset in two halves called first and second arrays. It reads the dataset into the arrays, and directly computes the distance

between each pair of object.
For each object t in the first array, a count of its D-neighbors is maintained. Counting stops for a particular object whenever the number of D-neighbors exceeds M. The algorithm is showed in the appendix [A].
Algorithm NL avoids the explicit construction of any indexing structure, and its complexity is of order (kN^2).
Compared to a tuple-by-tuple brute force algorithm that pays no attention to $I/O's$, Algorithm NL is superior because it tries to minimize $I/O's$.
One of the weaknesses in these density and distance measures is their inability to handle data sets with regions of different densities. Also, for these methods to detect dense anomaly clusters, k has to be larger than the size of the largest anomaly cluster.
This creates a problem in defining the appropriate k to use, keeping in consideration that a large k increases the computation substantially.

2.3 Classification Methods

The Density-Based Anomaly Detection approaches have several short-comings. Abe et al. in [1] point out two main drawbacks:

a. Do not provide a semantic explanation of why a particular instance has been flagged as an outlier;

b. Relatively high computational requirement, since nearest-neighbor methods need to store all or a large part of the past examples for effective classification of future examples.

These are potential obstacles in the implementation of this anomaly detection methods to real datasets.
Hence, an alternative approach to the anomaly detection is the classification based.
It is important to give some background about the classification methods which are then used in the Classification-Based Anomaly Detection.

2.3.1 Introduction to Classification Methods

Interesting applications of the classification methods in this field are provided by Abe et al. in [1] and by Breiman in [16].
The latter author has also worked on several books related to the classification methods of which propose a simple example in [12]:

Example 1 *The San Diego Medical Center measures* 19 *variables (blood pressure, age, weight, etc...) within the first* 24 *hours when a new patient with heart problems is admitted. The idea behind is to create a method which predicts, with high probability, who will not survive at least* 30 *days based on the data retrieved in the first* 24 *hours. This classification rule is summarized by the Figure [2] published by Breiman et al. in [17]:*

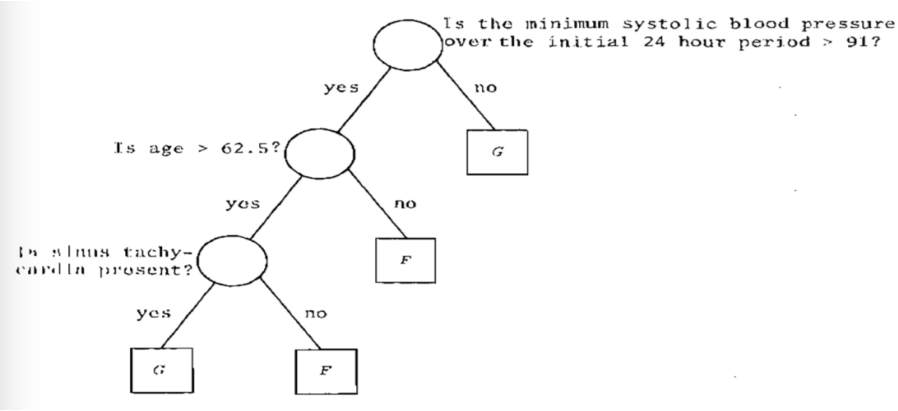

Figure 2: Classification Rule

where F means not high risk and G means high risk.
This rule classifies incoming patients as F or G depending on the yes-no answers to at most three questions. Given a set of measurements on a case or object, find a systematic way of predicting what class it is in.

In any problem, a classifier or a classification rule is a systematic way of predicting in which class a case will be.
To give a more precise formulation, arrange the set of measurement on a case in a preassigned order; i.e., take the measurement to be x_1,x_2,... x_n where x_1 is age, x_2 is blood pressure, etc.... Define the measurements (x_1,x_2,..., x_n) made on a case as the measurement vector $\mathbf{x}$ corresponding to the case. Take the measurement space χ to be defined as containing all possible measurement vectors.
In the heart attack study presented in [1], χ is a 19-dimensional space such as that the first coordinate x_1 (age) ranges, say, over all integer values from 0 to 200; the second coordinate, blood pressure, might be defined as continuously ranging from 50 to 150.

Multiple definitions of χ can be described, but any definition of the measurement space has the property that $\mathbf{x}$, the measurement vector, corresponding to a point in χ.
Suppose that the cases or objects fall into J classes, with classes 1,2,..., J and let ι be the set of classes; that is ι=(1,..., d). Then, a systematic way of predicting class membership is a rule that assigns a class membership in ι to every measurement vector $\mathbf{x}$ in χ.
Given any $\mathbf{x}\in\chi$, the rule assigns one of the classes (1,...,J) to $\mathbf{x}$. This can be summarized in the following definitions:

Definition 6 *A classifier or classification rule is a function $d(\boldsymbol{x})$ defined on χ so that for every $\boldsymbol{x}$, $d(\boldsymbol{x})$ is equal to one of the numbers* 1,2,...,J.

or in other words:

Definition 7 *A classifier is a partition of χ into J disjoint subsets A_1,..., A_j, χ=$\cup_j A_j$ such that for every $\boldsymbol{x}\in A_j$ the predicted class is j.*

Classifiers are built following a logic. Indeed, in systematic classifier construction, past experience is summarized by a learning sample which consists of the N cases observed in the past together with their classification.

Definition 8 *A learning sample consists of data $(x_i, j_1), \ldots, (x_n, j_n)$ on N cases where $x_n \in \chi$ and $j_n \in \{1, \ldots, j\}$, $n = 1, \ldots, N$.*
The learning sample is denoted by ζ; i.e.:

$$\zeta = \{(x_i, j_1), \ldots, (x_n, j_n)\} \tag{2.2}$$

Breiman in [12] distinguish two general types of variables that can appear in measurement vector:

Definition 9 *A variable is called ordered or numerical if its measured values are real numbers. A variable is categorical if it takes values in a finite set not having any natural ordering.*

A categorical variable, for instance, could take values in the set {red, blue, green}. On the other hand, the medical data for a hospitalized patient such as blood pressure and age are ordered variables. The latter is a typical example where the data structure is standard as a fixed set of variables is measured in each case (or day).

Definition 10 *If all measurement vectors x_n are of fixed dimensionality, we say that the data have standard structure.*

Depending on the problem, the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem.
In the first case, to determine if an object is in one class rather than another. In the second case, the final goal is to understand what variables drive a specific phenomena. These two goals are not mutually exclusive.
Hence, an important criterion for a good classification procedure is not only related to the production of accurate classifiers, but it is also to provide insights and understanding about the predictive structure of the data.
The major guide that has been used in the construction of classifiers is the concept of the Bayes' rule.
This rule can be defined as follow:

Definition 11 *$d_B(x)$ is a Bayes rule if for any other classifier d(x),*

$$P(d_B(\chi) \neq Y) \leq P(d(\chi) \neq Y) \tag{2.3}$$

where $P(A, j)$ is the probability distribution followed by the data. Then the Bayes misclassification rate is

$$R_B = P(d_B(\chi) \neq Y) \tag{2.4}$$

If a prior class probabilities π_j with $j = 1, \ldots, J$ is defined as

$$\pi_j = P(Y = j) \tag{2.5}$$

and the probabiity distribution of the jth class measurement vectors by

$$P(A|j) = \frac{P(A, j)}{\pi_j} \tag{2.6}$$

under the assumption that χ is a M-dimensional euclidean space and for every j, $P(A|j)$ has the probability density $f_j(x)$;

$$P(A|j) = \int_A f_j(x)dx. \tag{2.7}$$

Then, the Bayes rule can be written as:

$$d_B(x) = \{x; f_j(x)\pi(J) = \max_i f_i(x)\pi(i)\} \tag{2.8}$$

and the Bayes misclassification rate is:

$$R_B = 1 - \int \max_j [f_j(x)\pi(j)]dx \tag{2.9}$$

2.3.2 Different methods for applying Classification

Among the classification methods, the three most commonly used procedures are:

a. Discriminant analysis [38];

b. Kernel density estimation [41];

c. K-th nearest neighbor [33].

attempt, in different ways, to approximate the Bayes' rule by using the learning sample ζ to get estimates of $f_j(x)$.
A stepwise version of linear discrimination is the most widely used method. It is not constrained by the normality assumption and provides enough insights into the data structure. Nevertheless, the form of classifier for the J class problem is difficult to interpret.
The kernal density estimation and kth nearest neighbor methods make minimal assumptions about the form of the underlying distribution. But there are common drawbacks:

a. There is not simple way to handle categorical variables and missing data;

b. They are computationally expensive as classifiers; ζ must be stored, the interpoint distances and $d(x)$ recomputed for each new point x;

c. They do not give enough information about the data structure.

These problems that could not be handled in an easy way by any of the above-mentioned methods have alimented the experimentation of new methods. The use of classification trees is one of those.
Binary tree structured classifiers, are constructed by repeated splits of subsets of χ into two descendant subsets, beginning with χ itself.
This process, for a hypothetical six-class tree, is showed by the figure [3]:
χ_2 and χ_3 are disjoint, with $\chi = \chi_2 \cup \chi_3$. The same is valid for χ_4, χ_5 and so on.
The subsets which are not split are in a rectangular box and are called terminal nodes.
Each of those is designated by a class label and two or more terminal nodes could have the same class label. The partition corresponding to the classifier is got by putting together all terminal nodes corresponding to the same class.
For example: $A_4 = \chi_6 \bigcup \chi_{17}$.
The entire construction of a tree depends on three elements:

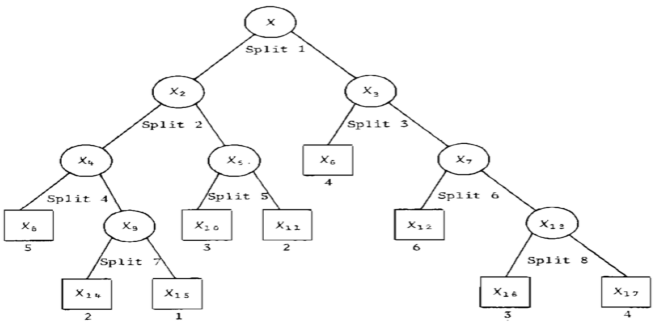

Figure 3: Classification Tree

a. The selection of the splits;

b. The decisions when to declare a node terminal or not;

c. The assignment of each terminal node to a class.

These items are well, extensively (and formally) explained by Breiman in [12].

2.3.3 Classification-Based Anomaly Detection

The classification methods briefly introduced in [2.3.2], have been extensively used in anomaly detection.
Among the others interesting application of the Classification-Based Anomaly Detection are showed in [1] and [16].
In [1], the authors reduce the outlier detection to classification and invoke the technique of active learning to the reduced classification problem. This approach has two benefits:

a. Selective sampling based on active learning is able to provide improved accuracy for outlier detection;

b. The use of selective sampling provides the data scalability that is needed for typical applications of outlier detection. This makes possible to handle very large data sets with low computational requirements.

They show how this approach outperforms either other methods based on the reduction to classification but using standard classification methods, such as bagging and boosting respectively well illustrated in [24], [13]

and [35] [36] [34] [65], or LOF shown in [18] which has been extensively studied as a method for outlier detection.
Several machine learning methods are suitable for anomaly detection. However, supervised algorithms are more constraining than unsupervised methods as they need to be provided with a labeled dataset which can also affect the efficiency of supervised algorithms [47].
On the other side, unsupervised algorithms make use of unlabeled data to assign a score to each sample, representing the observation normality. Binary segmentations can be further made by thresholding the scores [25].
The presented model utilizes a non-standard unsupervised learning method. Assuming that data are drawn from some probability distribution U on an instance space χ.
The goal in this model is to choose a good partition π of the space χ. The partition π divides the space χ into two subspaces which are called, π and $\chi - \pi$.
A 'good' partition π contains 'most' of the points while minimizing the size of π.
The error of the unsupervised learning is defined as:

$$e_{U,B}(\pi) = \frac{1}{2}(Pr_{x\sim U}(x \notin \pi) + Pr_{x\sim B}(x \in \pi)) \tag{2.10}$$

taking into account that B has been used to denote the 'background' distribution over X, the goal is to find a 'small' (B) set π which contains 'most' of the data (U).
A supervised learning model, using classification, is connected to the above-mentioned structure without any assumptions on the target concept.
Assuming a distribution D over the input space χ and the binary output space Y=$\{0, 1\}$, then a classifier $h : X \to Y$ with a small true error rate can be find as follow:

$$e_D(h) \equiv Pr_{x,y\sim D}(h(x) \neq y) \tag{2.11}$$

This connection enable to transfer much of the classification theory to this unsupervised learning model. In particular, deciding whether a point is drawn from a distribution B or U becomes a classification problem. Once the models are linked, a selective sampling mechanism, which normally suits only supervised learning problems(such as classification), based on a particular type of active learning methodology developed in [53] is applied.
This approach can be called as ensemble-based minimum margin active learning. The approach to outlier detection developed in [1] works iteratively, yielding a classifier in each iteration by feeding a sub-sample of the

input data set obtained by selective sampling to the given classification learner. This process is well explained by the figure [4] in [1].

Active-Outlier(Learner A, Samples S_{real}, count t, threshold θ, underlying distrubtion B)

1. Generate a synthetic sample, S_{syn}, of size $|S_{real}|$, according to B.
2. **Let** $S = \{(x, 0)|x \in S_{real}\} \cup \{(x, 1)|x \in S_{syn}\}$.
3. **For** $i = 1$ **to** t **do**
   (a) Let $M(x) = margin(\{h_0, ..., h_{i-1}\}, x)$.
   (b) Scan through S and obtain $S' =$ **by rejection sampling from** S **with sampling probability:**
   $$gauss(i/2, \sqrt{i}/2, (i + M(x))/2)$$
   (c) **Let** $h_i \equiv A(S')$
   (d) **Let** $\epsilon_i \equiv$ error rate of h_i on S'.
   (e) **Set** $\alpha_i = \log \frac{1-\epsilon_i}{\epsilon_i}$.
4. **Output** $h(x) = \textbf{sign}\left(\sum_{i=1}^{t} \alpha_i h_i(x) - \theta(\sum_{i=1}^{t} \alpha_i)\right)$

Figure 4: Outlier detection method using active learning [1]

The main challenges in the generation of this algorithm are generated by:

a. The choice of the underlying distribution B, or how to generate the synthetic sample S_{syn}. They choose two alternatives the uniform distribution within a bounded sub-space and product distribution of the marginals;

b. The definition of margin. Which is defined as:

$$margin(F, x) \sum_{\hat{f} \in F} \hat{f}(1|x) \sum_{\hat{f} \in F} \hat{f}(0|x); \tag{2.12}$$

c. The choice of ensemble weights α_i. Where they adopt the weighting scheme of AdaBoost developed in [35];

d. Normalizing constant for sampling probability. When rejection sampling results too small in sample sizes the sampling probability w_i, in each iteration, is multiplied by a normalizing constant $\frac{r}{w_i}$ where r is a pre-specified fraction. In this way the same fraction of examples is expected in each iteration.

The datasets used by Lazarevic and Kumar in [50] allow to compare their method either against other methods that are also based on the reduction to classification such as bagging and boosting, or against well-known outlier detection methods (density-based) such as LOF method [18] and Feature Bagging [50].
Abe et al. in [1] highlight how Active-Outlier is performing consistently close to best in all cases.
Nevertheless, most of the times the best is achieved by bagging, while Boosting, given the context, does not work well. The latter example confirms what is already known literature, in fact, Boosting tends to put too much weight on the incorrectly labeled examples in case of high noise.
This noise is generated during the reduction to classification problem where artificial samples are introuced.
Looking at the comparison with the outlier detection methods, LOF and Feature Bagging, the Outlier Detection by Active Learning achieves an AUC that is either equivalent or significantly better than the Feature Bagging, and outperforms LOF with a significant margin in all cases.
A different application of the classification methods in the anomalies detection is provided by Breiman in [16].
He explains how bagging, random split selection and adaptive bagging have inducted to significant improvements in classification accuracy:

a. **Bagging** in [13], where to grow each tree a random selection (without replacement) is made from the examples in the training set;

b. **Random split selection** in [24], where at each node the split is selected at random from among the K-best splits;

c. **Adaptive bagging** in [15], where generates new training sets by randomizing the outputs in the original training set.

In all of these procedures the $k-th$ tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_1, \ldots, \Theta_{k-1}$, but with the same

distribution; and a tree is grown using the training set and Θ_k, resulting in a classifier $h(\mathbf{x}, \Theta_k)$ where $\mathbf{x}$ is an input vector.
The nature and the dimensionality of Θ depends on how the tree is constructed (bagging, random split selection, etc. . .).
This procedure where ensembles of trees are generated and the most popular class is voted is called random forest and can be defined as follow:

Definition 12 *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(\boldsymbol{x},\Theta_k),\ k = 1,\dots\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\boldsymbol{x}$.

As highlighted by Biau and Scornet in [7], the popularity of the random forests derives from the fact that they can be applied to a wide range of prediction problems and have few parameters to tune. Aside from being simple to use, the method is generally recognized for its accuracy and its ability to deal with small sample sizes and high-dimensional feature spaces. For having a better understanding of the random forest is good to have a deeper view in their properties:

a. As the number of trees increases, for almost surely all sequences $\Theta_1,\dots,PE^*$ converges to:

$$P_{\mathbf{X},Y}(P_\Theta(h(\mathbf{X},\Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X},\Theta) = j) < 0). \tag{2.13}$$

where PE^* represents the generalization error and it is equal:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X},Y) < 0) \tag{2.14}$$

where mg($\mathbf{X}$,Y) is margin function which measures the extent to which the average number of votes at $\mathbf{X}$,Y for the right class exceeds the average vote for any other class. The larger the margin, higher is the confidence in the classification.

b. **Strength and correlation.** For random forests, an upper bound can be derived for the generalization error in terms of two parameters that are measures of how accurate the individual classifiers are and of the dependence between them. The upper bound for the generalization error is given by:

$$PE^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{2.15}$$

where $\bar{\rho}$ is the mean value of the correlation and s represents the strength of the set of classifiers $\{h(\mathbf{x}, \Theta)\}$ which can be written as:

$$s = E_{\mathbf{X},Y} mr(\mathbf{X}, Y) \tag{2.16}$$

Putting $c = \bar{\rho}(1 - s^2)$, a new ratio is obtained $\frac{c}{s^2}$.
As it represents the upper bound for the error, a smaller ratio, between average of the correlation and strength, will reflect in a better accuracy of the classifier.
This theorem is formally showed in [16].

In the latter article, Breiman explains as for improving accuracy, the randomness injected has to minimize the correlation, $\bar{\rho}$, while maintaining strength.
He introduces forests which randomly selected inputs or combinations of inputs at each node to grow each tree which have the following characteristics:

a. Its accuracy is as good as Adaboost and sometimes better, which are considered the most accurate;

b. It is relatively robust to outliers and noise [2];

c. It is faster than bagging or boosting;

d. It gives useful internal estimates of error, strength, correlation and variable importance;

e. It is simple and easily parallelized.

Besides, in Adaboost selects the distributions of the weights on the training set while in the usual random forest, the distribution of the random vectors does not depend on the training set
In his algorithm, he uses bagging in tandem with random feature selection where each new training set is drawn, with replacement, from the original training set. Then a tree is grown on the new training set using random feature selection. The trees grown are not pruned.
The reason behind the use of the bagging are the following:

a. To enhance accuracy when random features are used;

[2] Dietterich [24] shows that when a fraction of the output labels in the training set are randomly altered, the accuracy of Adaboost degenerates, while bagging and random split selection are more immune to the noise.
Since some noise in the outputs is often present, robustness with respect to noise is a desirable property.

b. To give ongoing estimates of the generalization error of the combined ensemble of trees(PE*), strength and correlation. These estimates are done out-of-bag[3] as explained in [70] and [14].

Breiman presents two procedures, Forest-RI and Forest-RC, and compare them with Adaboost.
The so-called Forest-RI is a procedure where a random forest with random features are formed by selecting at random, at each node, a small group of input variables to split on.
The size of the group (number of features) is fixed at priori and it is denoted by F, while the number of input is denoted by M .
He compared this procedure with Adaboost and formulates the following considerations:

a. Growing 100 trees in random forests is many times faster than growing 50 trees based Adaboost given the huge set of inputs needed in this procedure;

b. The error rates using random input selection compare favorably with Adaboost.

Nevertheless, when we have few inputs, or in other words M is small, and $F \simeq M$, then the results could be affected by high correlation.
The so-called Forest-RC procedures consists of defining more features by taking random linear combinations of a number of the input variables. This linear combination are defined by L which represents the number of variables to be combined.
At a given node, L variables are randomly selected and added together with coefficients that are uniform random numbers on $[1, 1]$.
F linear combinations are generated, and then a search is made over these for the best split.
Overall the generalization error compares more favorably to Adaboost than Forest-RI.
Besides, it is relevant to look at the effect of strength and correlation on the generalization error.
Breiman in [16] shows that for small datasets, past a small value of F (around 4), adding more inputs or features does not help because the strength remains constant and the correlation continues to increase.

[3] In each bootstrap training set, about one-third of the instances are left out. Therefore, the out-of-bag estimates are based on combining only about one-third as many classifiers as in the ongoing main combination

In other words, since the correlations are slowly but steadily increasing, the lowest error occurs when only a few inputs are utilized.
On the other hand, he highlights that for large datasets, either the correlation or the strength have a small but steady increase while the error rates show a slight decrease.
Hence, in presence of more complex data sets, the strength continues to increase longer before it is flatten out.
These results indicate that better random forests, which present a lower generalization error, have lower correlation between classifiers and higher strength.

2.4 Clustering-Based Anomaly Detection

He at al. present in [45] a different approach to the outlier detection called Clustering-Based Anomaly Detection.
As we did in this thesis, they start from the classical definition of outliers and they use the one proposed by Hawkins in [44]:

Definition 13 *An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*

The motivation of their work was driven by two reasons:

a. Existing algorithms, as for example the ones showed in [49], [62] and [18], for outlier detection involve high computation costs which is a big limitation in the era of the high dimensions;

b. Clustering algorithms, as ROCK in [39], C2P in [58] and DBSCAN in [28], have focused on the large patterns without specifically paying their attention on outlier detection, in fact the anomalies are seen as noise.

Hence, beyond to try to cover one of the aspects that has been always criticized in literature to the application of clustering-based methods in anomaly detection, because focused on large patterns instead of small ones where anomalies should be found, they introduce Clustering-Based Anomaly Detection which has the clear advantage of using one process for identifying large patterns and outliers in one go, without implement multiple processes and algorithms for different data mining tasks.
Nevertheless, as underlined in [2.1], different approaches to the anomaly detection can bring to a different outliers definition. In this case, He at al.

in [45] present a novel definition of outlier such as the cluster-based local outlier:

Definition 14 *Let $A_1, \ldots, A_m$ be a set of attributes with domains $D_1, \ldots, D_m$ respectively. Let the dataset D be a set of records where each record $t : t \in D_1 \times \cdots \times D_m$. The results of a clustering algorithm executed on D is denoted as: $C\{C_1; C_2; \ldots; C_k\}$ where $C_i \cap C_j = \emptyset$; and $C_1 \cup C_2 \ldots C_k = D$. The number of clusters is k.*

Leaving free the possibility to chose a clustering algorithm for partitioning the dataset into disjoint sets of records, they identify if a cluster is large or small in the following way:

Definition 15 *Suppose $C = \{C_1; C_2; \ldots; C_k\}$ is the set of clusters in the sequence that $\|C_1\| \geq \|C_2\| \geq \cdots \geq \|C_k\|$.*
Given two numeric parameters α and β, we define b as the boundary of large and small cluster if one of the following formulas holds:

$$(\|C_1\| + \|C_2\| + \cdots + \|C_b\|) \geq \|D\| * \alpha; \tag{2.17}$$

$$\frac{\|C_b\|}{\|C_{b+1}\|} \geq \beta. \tag{2.18}$$

Then, the set of large cluster is defined as:

$$LC = \{C_i | i \leq b\} \tag{2.19}$$

and the small cluster as:

$$SC = \{C_j | j \geq b\} \tag{2.20}$$

[4]

Finally at this stage, Cluster-based local outlier factor (CBLOF) can be defined.

[4] Equations [2.17] and [2.18] give quantitative measure to distinguish large and small clusters. As showed in [45], formula [2.17] considers the fact that most data points in the data set are not outliers. Therefore, clusters that hold a large portion of data points should be taken as large clusters. For example, if α is set to 90%, we intend to regard clusters contain 90% of data points as large clusters. Formula [2.18] considers the fact that large and small clusters should have significant differences in size. For instance, it is easy to get that, if we set β to 5, the size of any cluster in LC is at least five times of the size of the cluster in SC.

Definition 16 *Suppose* $C = \{C_1; C_2; \ldots; C_k\}$ *is the set of clusters in the sequence that* $\|C_1\| \geq \|C_2\| \geq \cdots \geq \|C_k\|$. *For any record* t*:*

$$CBLOF(t) = \begin{cases} \|C_i\| * min(distance(t, C_j)) \\ where\ t \in C_i, C_i \in SC\ and\ C_j \in LC\ for\ j = 1\ to\ b \\ \|C_i\| * distance(t, C_i)) \\ where\ t \in C_i\ and\ C_i \in LC \end{cases}$$

As result, Cluster-based local outlier factor of a record is determined by the size of its cluster, and the distance between the record and its closest cluster, either small or large, which provides importance to the local data behavior.
He at al. in [45] propose two algorithms for the computation of distance between the record and the cluster which can determine the degree of a record's deviation.
The Squeezer and the FindCBLOF algorithms are respectively described in the appendix [B] and [C].

2.5 Isolation-based Anomaly Detection

Isolation-based Anomaly Detection presented by Liu et al. in [51] and [52] is a newer approach, and less explored in literature, to the anomaly detection with respect to Density-Based Anomaly Detection, Classification-Based Anomaly Detection and Clustering-Based Anomaly Detection.
Once again the definition of outliers plays a key role, in fact, starting from the concept that anomalies are data points that are few and different, they are susceptible to a mechanism called isolation where the term isolation stands for 'separating an instance from the rest of the instances'.
In general, an isolation-based method measures individual instances' susceptibility to be isolated; and anomalies are those that have the highest susceptibility.
They propose a method called Isolation Forest (iForest) which has several properties:

a. The characteristic of isolation trees enables them to exploit subsampling to an extent that is not feasible in existing methods. This property helps in achieving a low linear time-complexity and in dealing with the effects

of swamping[5] and masking[6];

b. Does not utilize distance or density measures to detect anomalies. This tackles a major computational cost of distance calculation which is penalizing the distance-based and density-based methods;

c. Linear time complexity with a small constant and a minimal memory requirement; it is an algorithm with constant training time and space complexities;

d. The capacity to scale up to handle extremely large data size and high-dimensional problems with a large number of irrelevant attributes .

This methods only requires a small subsampling size to achieve high detection accuracy with high efficiency where the different height limits are used to cater for anomaly clusters of different density.
Besides, they empirically show that iForest outperforms $ORCA$, one-class SVM, LOF and $RandomForests$ in terms of AUC, processing time, and it is robust against masking and swamping effects[7] which break down many anomaly detectors.
Indeed, iForest is able to build a model by using multiple sub-samples which reduce those effects and these small size sub-samples build better iTrees than from the entire data set, they have fewer normal points 'interfering' with anomalies which makes anomalies easier to isolate.
Furthermore, iForest does not present issues with an increasing value of k, indeed it also works well in high dimensional problems containing a large number of irrelevant attributes, and when anomalies are not available in the training sample.
Most of the anomaly detection approaches, including density-based methods, their generalization called distance-based, provided by Knorr and Ng in [49], classification-based methods and clustering-based ones, construct a profile of normal instances, then identify anomalies as those that do not conform to the normal profile.
Mostly for classification and clustering techniques, the anomaly detection capabilities are a 'side-effect', in other words these algorithms are originally

[5] Swamping refers to situations where normal instances are wrongly identifying as anomalies. It happens when the number of normal instances increases or they become more scattered.

[6] Masking is the existence of too many anomalies concealing their own presence. It happens when anomaly clusters become large and dense.

[7] The problems of swamping and masking have been studied extensively in anomaly detection, i.e Murphy in [57]

designed for identifying the large patterns instead of the small patters or outliers. Exceptions to this general statement are provided in [2.3.2] and [2.4].
This leads to two major short-comings:

a. In general, these approaches underperform because they are not designed for detecting anomalies. The outcome shows too many false alarms, where we intend normal instances identified as anomalies, or too few anomalies being detected;

b. Many existing methods work well with low dimensional data and small because of the legacy of their original algorithms.

The approach proposed by Liu et al. in [51], [52] detects anomalies purely based on the concept of isolation without employing any distance or density measure.
The iForest algorithm takes advantage of two quantitative properties of anomalies:

a. They are the minority, consisting of few instances;

b. They have attribute-values that are very different from those of normal instances.

In other words, anomalies are few and different.
Besides, for implementing the isolation, a binary tree structure, called isolation tree (iTree), which can be effectively constructed to isolate instances has been used.

Definition 17 ***Isolation Tree.*** *Let T be a node of an isolation tree. T is either an external-node with no child, or an internal-node with one test and exactly two daughter nodes (T_l, T_r). A test at node T consists of an attribute q and a split value p such that the test $q < p$ determines the traversal of a data point to either T_l or T_r.*
Let $X = \{x_1, \ldots, x_n\}$ be the given data set of a d-variate distribution. A sample of ψ instances $X' \subset X$ is used to build an isolation tree (iTree). We recursively divide X' by randomly selecting an attribute q and a spit value p, until either the node has only one instance or all data at the node have the same values.
An iTree is a proper binary tree, where each node in the tree has exactly zero or two daughter nodes. Assuming all instances are distinct, each instance is isolated to an external node when an iTree is fully grown,

in which case the number of external nodes is ψ and the number of internal nodes is $\psi - 1$; the total number of nodes of an iTrees is $2\psi - 1$; and thus the memory requirement is bounded and only grows linearly with ψ.

In this tree structure, the instances are recursively partitioned, these trees produce noticeable shorter paths for anomalies since:

a. In the regions occupied by anomalies, less anomalies result in a smaller number of partitions;

b. Instances with distinguishable attribute-values are most likely separated early in the partitioning process.

Since recursive partitioning can be represented by a random tree structure, the number of partitions required to isolate a point is equivalent to the traversal of **path length** from the root node to a terminating node. Path length can be defined as follow:

Definition 18 ***Path Length.*** *$h(x)$ of a point x is measured by the number of edges x traverses an iTree from the root node until the traversal is terminated at an external node.*

Hence, given the susceptibility to isolation, anomalies are most likely isolated closer to the root of an iTree which means that they have short average path lengths on the iTrees.
In other words, when a forest of random trees collectively produce shorter path lengths for some particular points, they are highly likely to be anomalies.
On the other hand, normal points are most likely isolated at the deeper end of an iTree.
The task of anomaly detection is to provide a ranking that reflects the degree of anomaly. Using iTrees, the way to detect anomalies is to sort data points according to their average path lengths; and anomalies are points that are ranked at the top of the list.
This concept of path, helps in building path-length-based isolation approach which differs from density and distance based ones.
Besides, Liu et al. in [51] and [52] highlight that the isolation approach tackles some of the short-comings of density-based and distance-based approaches.
For example in the density-based is assumed that "Normal points occur in dense regions, while anomalies occur in sparse regions", nevertheless high

density do not always imply normal instances as well as low density do not always imply anomalies.
One of the most used density-based approach such as LOF, showed in [18], for solving some issues utilize a concept of local, nevertheless points with high density could be anomalies in the global context of the entire data set. From an helicopter view, the isolation method is similar to a density measure or a distance measure, in fact isolation ranks scattered outlying points higher than normal ones.
However, path-length-based approach grows tree in an adaptive context where each partitioning is different, from the first partition (the root node) to the last partition (the leaf node), where the approach take into account the different context, respectively the entire data set and local data-points. On the other hand, the most common density ($k-nn$) and distance ($k^{th}nn$) approaches only concern the local context taking care of the k-neighbors and failing the analysis of the entire data (global perspective).
In other words, comparing path, density and distance approaches highlights how path-length-based is able to detect either clustered or scattered anomalies while the other approaches can only capture the latter.
The path-length-based isolation anomaly detection approach has two-stage process:

1. A training stage which builds isolation trees using sub-samples of the given training set

2. An evaluation stage which tests instances through isolation trees to obtain an anomaly score for each instance.

In practice, the iForest algorithm builds an ensemble of iTrees for a given data set; This algorithm has two training parameters:

a. Number of trees to build (t);

b. Subsampling size (ψ).

and one evaluation parameter which is the tree height limit during evaluation.
In the first stage, the training one, $iTrees$ are constructed by recursively partitioning the sub-sample X' until all instances are isolated. Each $iTree$ is constructed using a sub-sample X' randomly selected without replacement from X, where $X' \subset X$.
The **subsamplig size**, ψ, controls the training size. Starting from the assumption of anomalies given in [52] the anomalies are 'few' and 'different',

so a small subsampling size is enough for *i*Forest to distinguish anomalies from normal points. This means that once the ψ desired value is reached, a further increase of ψ is not needed because it increases the processing time and memory size without any further gain in detection accuracy.
In their empirical studies they observe that $\psi = 256$ is enough to perform anomaly detection.
With regard to the **Number of trees** (t), it controls the ensemble size. In this case, the empirical work shows that the path lengths usually converge well before $t = 100$.
The **evaluation stage** derives a single path length $h(x)$ by counting the number of edges e from the root note to an external node instance x traverses through an $iTree$.
When the traversal reaches the threshold height value ($hlim$), then the return value is e plus an adjustment $c(Size)$. This adjustment accounts for estimating an average path length of a random sub-tree which could be constructed using data of $Size$ beyond the tree height limit.
At the end, when $h(x)$ is computed for each tree of the ensemble, an anomaly score is calculated.
In isolated-based anomaly detection context, it has to be decided if an isolated data cluster is abnormal or not.
*i*Forest is able to detect in either case by changing the tree height limit parameter at the evaluation stage. Note that adjusting the height limit at evaluation stage does not alter the trained model and it does not require a re-training of the model.
In the normal usage of iForest, the default value of evaluation height limit is set to maximum, i.e. $\psi - 1$, so that the anomaly score has the highest granularity.
Either for comparison or visualization purpose, this anomaly score has to be normalized. Otherwise, the path lengths from models of different subsampling sizes cannot be directly compared.
Starting from an article of Preiss and R [60] and their analysis on the Binary Search Tree (BST) and highlighting that *i*Trees have an equivalent structure of the BST, Liu et al. in [52] estimate the average path length of *i*Tree as:

$$c(\psi) = \begin{cases} 2H(\psi - 1) - \frac{2(\psi-1)}{n} & for\ \psi > 2, \\ 1 & for\ \psi = 2, \\ 0 & otherwise. \end{cases}$$

where $H(i)$ is the harmonic number and it can be estimated by $ln(i) + 0.5772156649$ (Euler's constant). As $c(\psi)$ is the average of $h(x)$ given ψ, it

can be used to normalize $h(x)$.
The anomaly score s of an instance x can defined as:

$$s(x, \psi) = 2 - \frac{E(h(x))}{c(\psi)} \tag{2.21}$$

where $E(h(x))$ is the average of h(x) from a collection of iTrees,
Three special cases of the anomaly score are enumerated below:

a. if $E(h(x)) \to 0$, $s \to 1$;

b. if $E(h(x)) \to \psi$, $s \to 0$;

c. if $E(h(x)) \to c(\psi)$, $s \to 0.5$.

The relationship of expected path length $E(h(x))$ and anomaly score s is illustrated in the following figure [5].
Using s as anomaly score the following statements can be made:

a. when s is very close to 1, then the instances are definitely anomalies;

b. when s is much smaller than 0.5, most probably these are normal instances;

c. when all the instances return $s \sim 0.5$, then the entire sample does not have any anomalies.

The figure [6] produced by Liu et al. in [52] clearly visualize the above-mentioned process: Using the contour, the potential anomalies are clearly visualized for $s > 0.6$.

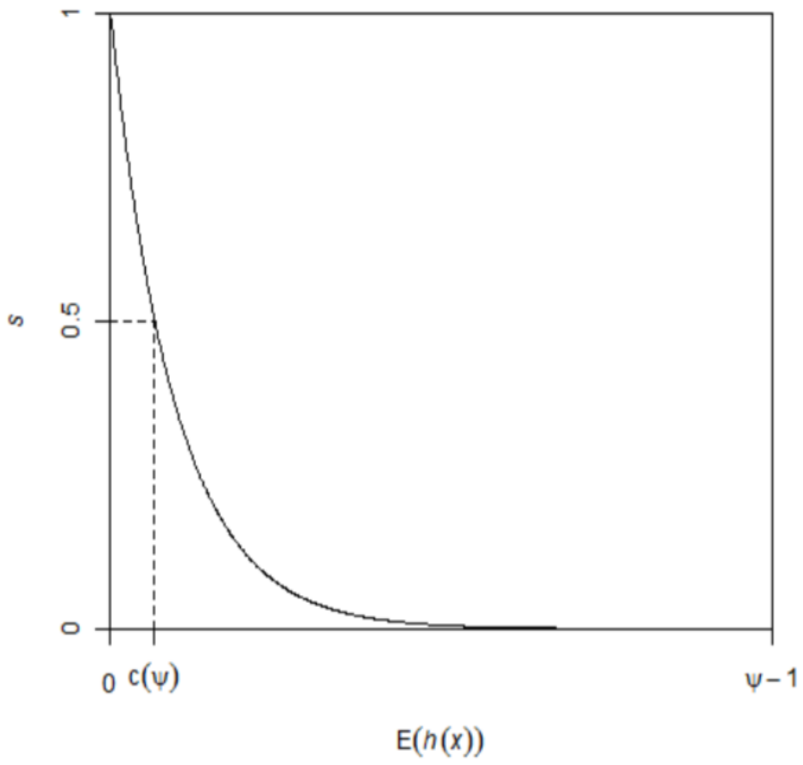

Figure 5: The relationship of expected path length E(h(x)) and anomaly score s

.

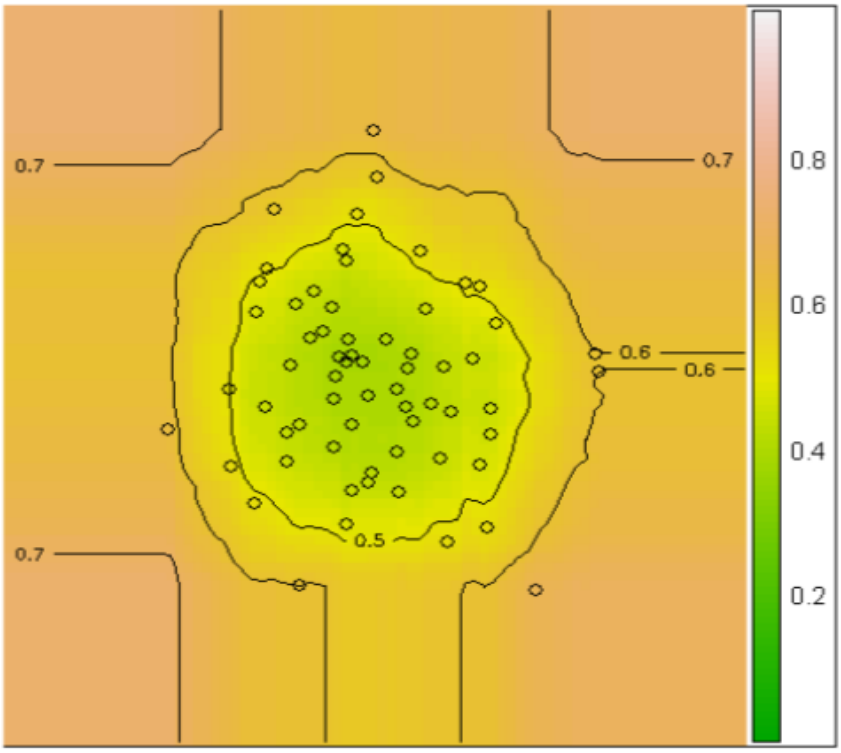

Figure 6: Anomaly score contour of iForest

Chapter 3

Testing algorithms on artificial sample data

3.1 Testing algorithms on random sample data

Using as starting point this literature state of the art seen in [2.5], we want to test these algorithms on artificial sample datasets, understand their properties and finally, in the chapter [4], apply those on real credit risk dataset. Among the methods and algorithms shown in [2.5], Domingues et al. underlines in [25] that Local Outlier Factor (LOF) outperforms several algorithms when applied on real-world datasets for outlier detection, which makes it a good candidate for this analysis.
On the other hand, Isolation Forest, previously seen in [2.5], introduced by Liu et al. in [51] and [52], which uses random forests to compute an isolation score for each data point, is granted by the authors to provide linear time complexity and, at the same time, they demonstrate that the Isolation Forest outlier detection performance is significantly better than LOF on real-world datasets.
This introduction makes clear our candidates for the testing purpose.
Indeed the test on the anomaly detection will be conducted using and comparing mainly 2 algorithms:

1. Distance-Based algorithm: Local Outlier Factor (LOF);

2. Isolation-Based algorithm: Isolation Forest (IF).

3.1.1 Introduction to the algorithms comparison

Always beneficial to quickly recall the given definitions of the algorithms enumerated in [3.2.1]:

1. **The Local Outlier Factor (LOF)** algorithms are well-known in literature as part of the Distance-Based family. These algorithms have been showed by Hawkins in [44] and defined in [4] as follow:

   (a) An object O in a dataset T is a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O The term $DB(p, D)$-outlier stands for a Distance-Based outlier.

   Nevertheless, for many applications, the situation is more complex and it becomes more meaningful to assign to each object a degree of being an outlier.
   To tackle this short coming the local outlier technique (LOF) has been introduced by Breunig et al. in [18] which computes a score (called local outlier factor) reflecting the degree of abnormality of the observations.
   This detects the local density deviation of a given data point with respect to its neighbors, spotting the samples that have a substantially lower density than their neighbors.
   The LOF score of an observation is equal to the ratio of the average local density of his k-nearest neighbors, and its own local density.
   In other words, a 'normal' instance is expected to have a local density similar to that of its neighbors, while 'abnormal' data are expected to have much smaller local density.
   In conclusion, LOF performs quite well even in datasets where abnormal samples have different underlying densities, since it does not try to answer the question how isolated the sample is, but it focuses on how isolated the sample is with respect to the surrounding neighborhood.
   Hence the following definition can be introduced:

   Definition 19 *For a given data point x, LOF computes its degree $d_k(x)$ of being an outlier based on the Euclidean distance d between x and its k_{th} closest neighbor n_k, which gives $d_k(x)=d(x, n_k)$. The scoring of x also takes into account for each of its neighbors n_i, the maximum between $d_k(n_i)$ and $d(x, n_i)$.*

2. **The Isolation Forest (IF)** is one of the most efficient way of performing outlier detection in high-dimensional datasets.

This algorithm leverages on the mechanism of isolation and on one of the most popular classification methods, Random Forest, introduced by Breiman in [16].
The Isolation Forest algorithm described in [2.5] can be summarized and defined with the following:

Definition 20 *The Isolation Forest (IF) algorithm 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Hence, it performs a recursive random splits on attribute values, generating trees able to isolate any data point from the rest of the data.*
The score of a point is then the average path length from the root of the tree to the node containing the single point, a short path denoting a point easy to isolate due to attribute values significantly different from nominal values.
In other words, the path length, averaged over a forest of such random trees is a measure of normality and our decision function, when a forest of random trees collectively produce shorter path lengths for particular samples, they have high probabilities to be anomalies.

3.2 Comparison of the Data Anomalies Detection algorithms

The ideal situation, in order to challenge the state of the art, we should be in the position to apply, test and compare these algorithms on a real dataset.
Nevertheless, we have 2 main problems which are stopping us from it:

1. Computational problems for $K > 2$;

2. Backtesting of the algorithms.

where K is the number of the futures that the algorithm has to take into account. In fact, high dimensional problems can be memory expensive.

3.2.1 Python settings for Local Outlier Factor algorithm

One of the key elements in the LOF algorithm is to choose the number of the k neighbors to consider which is one of the parameters to set in

the algorithm. In general, in order to choose this parameter, the below mentioned rules are followed:

1. k is greater than the minimum number of objects that a cluster has to contain, so that other objects can be local outliers relative to this cluster;

2. k is smaller than the maximum number of close by objects that can potentially be local outliers.

In practice, such information is generally not available.
On the other hand, $n\_neighbors = 20$ appears to work well as proxy.
The LOF algorithm can be called in Python using the $LocalOutlierFactor$ procedure.
Unfortunately, Python does not offer standard functions to predict and score results for the LOF algorithm as for Isolation Forest (e.g. $decision\_function$ and $score\_samples$).
Nevertheless, the results of the prediction can be achieved using a function such as $fit\_predictmethod$, while the abnormality scores of the training samples can be obtained through the $negative\_outlier\_factor\_$ attribute.
The anomaly score of each sample is called Local Outlier Factor and it measures the local deviation of density of a given sample with respect to its neighbors.
In the LOF context, each sample is defined as 'local' with respect to the k-nearest neighbors [1] and the anomaly score depends on how isolated, or in other words how distant, the object is with respect to the surrounding neighborhood.
The samples considered as outliers result to have a local density substantially lower than the local densities of their neighbors.

3.2.2 Python settings for Isolation Forest algorithm

As already underlined in [3.1.1], one efficient way of performing outlier detection in high-dimensional datasets is to use Isolation Forests.
This algorithm can be called in Python using the procedure $IsolationForest$.
This procedure contains a huge variety of parameters to set. In our case we have worked with the following:

1. **n_estimators** represents the number of base estimators in the ensemble;

[1] This gives some insights on how to use the parameter $n\_neighbors = 20$ and how changing it will impact the results of the algorithm

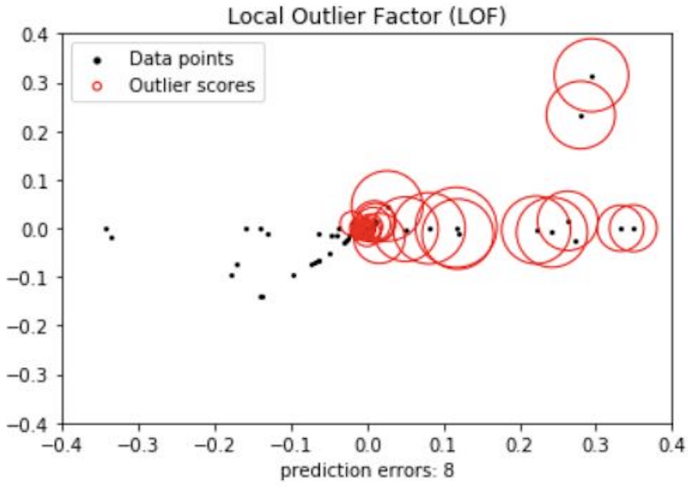

Figure 7: Local Outlier Factor

2. **contamination** is the amount of contamination of the data set used when fitting to define the threshold on the decision function, in other words the proportion of outliers expected in the data set;

3. **n_jobs** indicates the number of jobs to run in parallel for both fit and predict;

4. **max_samples** indicates the number of samples to draw from X to train each base estimator. When the parameter max_samples is bigger than the number of samples provided then all samples available will be used for building the trees which means that there will not be any sampling;

5. **max_features** is the number of features to draw from X to train each base estimator;

6. **behavior** is a parameter of the decision_function which can be set either as 'old' or 'new'. Setting this parameter as 'new' makes the decision_function change to match other anomaly detection algorithm API which will be the default behavior in the future;

7. **random_state** is the seed used by the random number generator.

It is interesting to understand how some specific attributes or methods works in this algorithm.

Regarding the attributes, it is interesting to shed more light on the attribute $offset\_$ which is used to define the decision function from the raw scores.
Starting from the definition of the $decision\_function$ for the IF algorithm in Python:

$$decision\_function = score\_samples - offset\_ \tag{3.1}$$

we observe that when the $behavior$ parameter is set to 'new' the $decision\_function$ becomes dependent on the $contamination$ parameter and, in this case, the offset is defined in such a way that the expected number of outliers is obtained.
In other words, 0 becomes the natural threshold to detect outliers in the training set, where scores lower than 0 are labelled as outliers.
On the other hand when the contamination parameter is set to 'auto', the offset is equal to -0.5 as the scores of inliers are close to 0 and the scores of outliers are close to -1.
In conclusion, if the $behavior$ parameter is set to 'old' the offset_ parameter is always equal to -0.5 and $decision\_function$ becomes independent from the contamination parameter.
Among the possible methods to set in the IF algorithms in Python, we have some interesting such as:

1. **decision_function(X)** which shows the average anomaly score of X of the base classifiers.
   The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest.
   In this case, the measure of 'normality' of an observation given a tree is the depth of the leaf containing this observation, which is equivalent to the number of splittings required to isolate this specific point.
   The rule to follow when applying this method is that lower is the score more 'abnormal' is the observation.
   In more detail, the negative scores represent outliers while the positive ones represent the inliers;

2. **fit_predict(X, y=None)** method performs the fit on X and returns labels for it.
   In this case, in order to facilitate the decision, the method gives back -1 for outlier observations and 1 for inliers ones.

3.2.3 Local Outlier Factor vs. Isolation Forest

The figure [8] shows the properties of the LOF and IF algorithms on 2D datasets.
In order to obtain these graphical results, the Python code is built according to the following steps:

1. Define the dimension of the sample;
2. Define a contamination factor. In other words, the percentage of data anomalies;
3. Define the outlier detection methods to use for the comparison;
4. Build clusters in the dataset using the function *make_blobs*;
5. Generate outliers and add them to the dataset;
6. Apply the algorithms;
7. Plot the results.

The length of our samples is equal to 30000 data points randomly generated where the noise is obtained through a uniform random variable.
The contamination of each dataset is 10%, which means that 3000 observations out of 30000 are anomalies.
The datasets are created in Python using one of the available techniques for random sample generation of artificial datasets with controlled size and complexity.
This function, for dataset creation called *make_blobs*, creates multi-class datasets by allocating to each class one or more normally-distributed clusters of points.
make_blobs allows to have a great control regarding the centers [2], standard deviations of each cluster and the number of features to take into account per each sample, indeed these parameters can be easily set in the the function.
In our case, the created datasets contain 1 or 2 regions of high density which helps in showing the properties of the algorithms when they have to cope with multimodal data.
In the first line of the graph [8], we have created a random dataset with a single center or in other words with an unique cluster.

[2] This represents the number of clusters.

It is already clear how better IF performs compared with the LOF algorithm since points which are not part of the cluster are wrongly identified as inliers.
The other 2 rows of the graph [8], represent a multimodal datasets which have 2 modes. In the third line of the graph [8], performance of the algorithms are quite uncertain.
While LOF continues to consider as inliers points which are out of the main clusters, the IF algorithm does not create a separation between the 2 clusters which may raise the suspect that the algorithm is not performing well for the points which are in the middle of the 2 centers[3].
Hence we have a training set which is not contaminated by outliers without any assumptions on the distribution of the inlying data.
Isolation Forest algorithm displays the decision thresholds (or boundaries) between inliers and outliers in black.
On the other hand, LOF does not have a predictive method, as mentioned in [3.2.1], in case of new data, so the decision boundary is not shown.
In this test, either IF or LOF perform well in presence of multi-modal datasets.
In particular, LOF performs well in these cases by design (or by definition), indeed the Local Outlier Factor algorithms only compares the score of abnormality of one point with the scores of its k-neighbors. In this case k has been set equal to 20.
This comparison is happening in a simplistic context, since the algorithms are applied on labelled data which creates a supervised learning problem where the model parameters, in this case, have been handpicked.
In absence of labelled data, the problem becomes unsupervised so model calibration can become an interesting challenge.
The main observation from the graph comparison is related to the highlighted[4] outliers.
Indeed, in the Isolation Forest graphs the demarcation seems to be much more clear, while in LOF some points have been identified as inliers even if they are outside of the main clusters.

[3] In the case of the third line of the graph [8], the function *make_blobs* is creating 2 random datasets with centers located in $(2, 2)$ and $(-2, -2)$

[4] Please note that the outliers are shown in blue while the inliers are in orange.

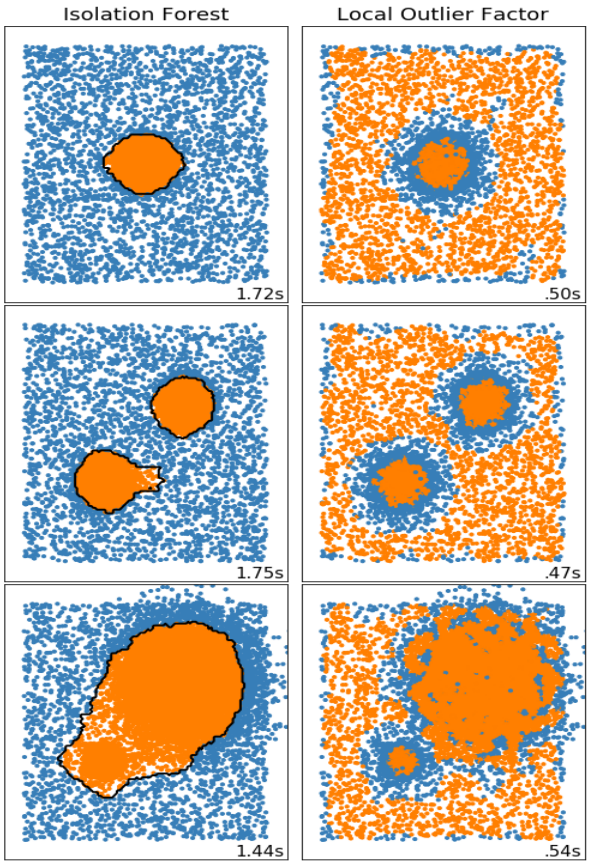

Figure 8: Isolation Forest vs. Local Outlier Factor

3.3 Information Criteria

While comparing different algorithms, it becomes relevant to understand the performance of those and understand how to measure their application on data.
In a binary decision problem, a classifier label is either positive or negative. In our application field, it is either an outlier or an inlier.
The detailed data produced by a classification method during the testing are counts of the correct and incorrect classifications from each class. The table which show the differences between the true and the predicted classes is called $confusion$ matrix.

Hence the $confusion$ matrix has 4 categories:

1. True Positives (TP): examples correctly labeled as positives;
2. False Positives (FP): correspond to negative examples incorrectly labeled as positive;
3. True Negatives (TN): refer to negatives correctly labeled as negative;
4. False Negative (FN) are positive examples incorrectly labeled as negative.

The table 3.1 shows an example of $confusion$ matrix.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 3.1: Confusion Matrix

Among the most used instruments in literature are known the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve which are graphical plots that illustrate the diagnostic ability of a binary classifier at various threshold settings.
In order to build and plot these curves, it is important to understand their components.
The ROC space plots on the x-axis the False Positive Rate (FPR) and on the y-axis the True Positive Rate (TPR).
In other words, ROC curve explain the trade-offs between true positive (benefits) and false positive (costs).
Trying to formalize the ROC curve, the FPR measures the fraction of negative examples that are misclassified as positive:

$$FPR(i) = \frac{FP_{(i)}}{TotalNegatives} \tag{3.2}$$

and TPR measures the fraction of positive examples that are correctly labeled:

$$TPR(i) = \frac{TP_{(i)}}{TotalPositives} \tag{3.3}$$

On the other hand, in the PR space, it has been plotted the Recall on the x-axis and Precision on the y-axis. If Recall is just a synonymous of the

TPR, the Precision measures that fraction of examples classified as positive that are truly positive:

$$PRECISION(i) = \frac{TP_{(i)}}{FP(i) + TP(i)} \tag{3.4}$$

3.3.1 The Precision-Recall curve

As explained by Davis and Goadrich in [22], for problems with heavily imbalanced class distributions where the positive class is more interesting than the negative class, since the negative class is the majority, the PR curves are particularly useful.
Indeed, the use of *precision* strongly penalizes methods raising FP even if those represent only a small proportion of the negative samples.
As underlined in [3.3], the PR curves show the tradeoff between precision and recall for different threshold.
This means that higher is the area under the curve higher will be the precision and recall, where high precision relates to a low FP, and high recall relates to a low FN.
Hence, a classifier with high scores is returning accurate results (high P) and it is capturing the majority of all positive results (high R).
In other words, if our algorithm shows high recall but low precision, it is an algorithm which returns many results, but most of its predicted labels are incorrect when compared to the training labels.
On the other hand, if the algorithm returns high precision but low recall, this turns out in very few results, but most of its predicted labels are correct when compared to the training labels.
In conclusion, the ideal algorithm has high precision and high recall, since it will return many results which are all labeled correctly.
Now, it is also important to understand what will influence P and R and how they are related between them.
Starting from the formula of P [3.5], shows that a lower classifier threshold may increase the denominator, since the number of results returned will be higher. This might turn out in a decreasing precision, since a lower threshold could increase FP.
Regarding R [3.3], this is not influenced by the classifier threshold.
This means that lowering the classifier threshold may increase R, by increasing the number of TP results, but it could also leave the R unchanged, while the P fluctuates.
Looking at these measures in the PR curve context, it means that a small

threshold reduction might considerably reduces precision with only a minor gain in recall.

3.3.2 The Receiver Operating Characteristic (ROC) curve

Spackman and Kent in [67] introduced the ROC curves in machine learning techniques evaluating different classification algorithms with those curves.
In order to draw a ROC curve, the measures introduced in [3.3] as TPR and FPR are needed.
The availability of these variables could be a problem on the real data application since in order to define those values the outliers must be known a priori.
TPR is also called $sensitivity$ while FPR is equal to $1 - specificity$, hence the ROC plot is sometimes called the sensitivity vs (1 - specificity) graph.
Then the ROC space is composed by all prediction result or instance of the confusion matrix.
In the figure [9], the left upper corner of the ROC space will contain the best possible prediction method, since that area of the ROC space represents the 100% sensitivity (absence of false negative) and the 100% of specificity (no false positives).
This is the reason why all the predictions falling at the coordinate (0,1) are results of a perfect classification.
Nevertheless, the ideal situation of a perfect classification is not often verified, so it becomes relevant to understand how to interpret the entire ROC space.
The diagonal line going from left bottom to the top rights corners is called line of no-discrimination and it represents all point given by a random guess (e.g. flipping the coins).
Indeed for a large sample size, a random classifier's ROC instance tends to the point (0.5,0.5) in case of a balanced coin. In other words, the diagonal divide good classification results (above the diagonal) from bad classification results (below the diagonal).

3.3.3 ROC vs. PR: Analysis and Interpretation of the Information Criteria

Several ways to summarize a precision-recall (PR) and ROC curves are known. These methods might lead to different results.
In this case the comparison of the ROC and PR curves are based on the area under the curve (AUC) of both metrics, respectively the ROC AUC

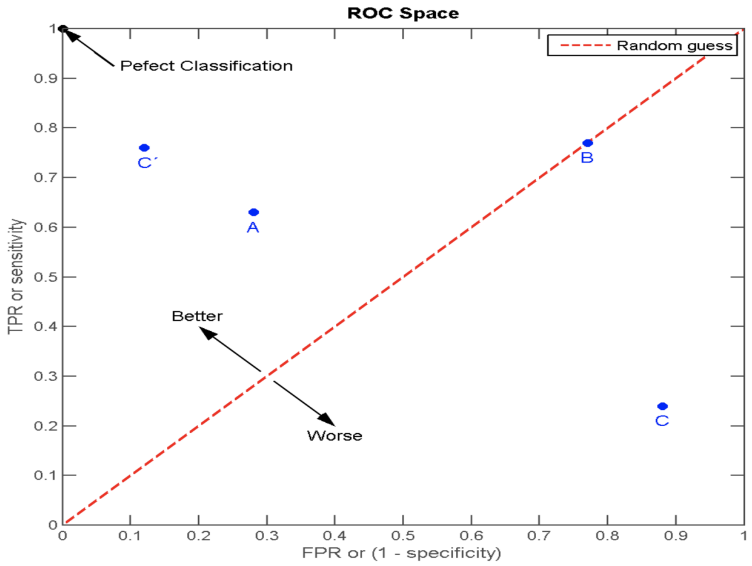

Figure 9: Representation of the ROC space

and the average precision (AP).
The most common metric for measuring the PR curve is the Average Precision (AP) as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{3.5}$$

where P_n and R_n are respectively the precision and recall at the n^{th} threshold. A pair (R_k, P_k) is referred to as an operating point.
In other words, the AP is the trapezoidal area under the operating points.
Regarding to the ROC curve we have to further formalize the TPR and FPR in order to understand how to use the ROC curve for measuring the performances of a classifier algorithm.
The outlier detection is a binary classification problem. In these cases, the class prediction for each instance is often made based on a continuous random variable X, which represents a "score". This score is calculated per each instance and compared with a threshold parameter T:

$$\begin{aligned} &If\ X > T\ then\ positive\ with\ a\ density\ probability\ f_1(x) \\ &\qquad else\ negative\ with\ a\ density\ probability\ f_0(x). \end{aligned} \tag{3.6}$$

Hence, we can re-write TPR and FPR as follow:

$$TPR(T) = \int_T^{\infty} f_1(x)dx \tag{3.7}$$

and

$$FPR(T) = \int_T^{\infty} f_0(x)dx \tag{3.8}$$

where T represents a varying parameter.
As explained by [31], the area under the ROC curve (or AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.
As showed by Brandley in [11], Hanley and McNeil in [43], to compare classifiers the ROC performance should be reduced to a single scalar value representing expected performance.
In other words, the area under the ROC curve is a common method to calculate the performances of a classifier.
Given X_1 as score for the positive instance and X_0 the negative ones, then:

$$\begin{aligned} AUC &= \int_{x=0}^{1} TPR(FPR^{-1}(x))dx = \int_{-\infty}^{\infty} TPR(T)FPR'(T)dT = \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T)f_1(T')f_0(T)dT'dT = P(X_1 > X_0) \end{aligned} \tag{3.9}$$

In conclusion, the AUC is related to the Gini coefficient by the following formula:

$$G_1 = 2AUC - 1 \tag{3.10}$$

where G_1 represents the Gini coefficient and it is calculated as:

$$G_1 = 1 - \sum_{k=1}^{n}(X_k - X_{k-1})(Y_k - Y_{k-1}) \tag{3.11}$$

Hence, following Hand et. al in [42], the AUC can be calculated as an average of a number of trapezoidal approximations.

3.4 Sensitivity analysis on algorithms' properties in literature

Additional details on the comparison of machine learning algorithms for outliers detection are provided by Domingues et al. who conduct a very

exhaustive and insightful analysis in [25] where they analyze 14 different machine learning techniques.
Amongst those techniques, the LOF and IF are included.
They study different properties such as robustness, complexity and memory usage of the algorithms on artificial datasets.
It is important to notice that this kind of tests were possible thanks to high computing power [5].
The mainstream is that IF over-performs all the other algorithms, including the LOF.
In more detail:

1. **Robustness**: They test the resistance of each algorithm to the curse of dimensionality where a fixed level of background noise is kept while increasing the dataset dimensionality.
   LOF does not perform well in noisy environments while good average results have been observed for IF;

2. **Complexity**: They focus on the computation and prediction time required by the different methods when increasing the dataset size and dimensionality.
   Increasing the number of features either IF or LOF show a stable training time and a good prediction time evolution.
   Hence, high dimensionality do not strongly affect these algorithms.
   On the other hand the number of samples has a strong impact on the training and prediction time of LOF which scale very poorly.
   LOF, together with other 4 algorithms, reaches the set timeout[6] of 24h for less than one million samples;

3. **Memory usage**: They measure the memory used by the corresponding running process before starting the algorithm and subtract it to the memory peak observed while running it.
   In order to do it they use the $memory\_profiler$ library for $Python$. For LOF and IF the memory does not increase for an increasing number of dimensions.
   The constant memory usage stays below 1MB. Different story is the increasing number of samples which has a much higher impact on the RAM consumption, mostly for LOF.

[5] The authors underline that the experiments were performed on virtual platform powered by an Intel CPU with 10 cores at 2.6 GHz and 256GB RAM

[6] The authors allowed till to 24h for training or prediction steps and they gave a timeout after this period of time

Indeed, LOF run out of memory for 193000 samples with a memory usage of 118GB.
IF scales much better to a higher numbers of samples since it stops at 10Million with a memory usage of 60GB RAM.
The huge amount of memory for LOF is mostly mostly caused by the use of a pairwise distance matrix which requires 76GB of RAM per each 100000 samples using 8 bytes per double precision distance.

Chapter 4

Application to Credit Risk Data

4.1 Data and Creation of the Samples

The importance of data in the past ten years is increased in an incredible way. Governments, Companies, Researchers and common people are trying to get more and more insights out of data.
The Data Scientist job title who is the person supposed to work with data combining business, IT and statistical knowledge has been defined as the sexiest job of the 21^{st} century and hundreds job positions related to it are popping up every day.
In a statistical environment, data represents a key role since it is the source that allows to develop, test and publish, either models or algorithms. The latter have the function to produce insights out of the data.
To give more background, will help to clarify the idea behind this research and to understand how data has been approached.
Firstly, the data sample will be extracted from the data warehouse. Then we have a data exploration step where we learn more about the attribute that we want to study (RWAs[1]).
Afterwards a different data sample[2], on which the models will be applied, is created.
Last but not least, the comparison of the Isolation Forest with other algo-

[1] RWAs have been taken as a key measure since they represent an aggregated measure of different risk factors. An overview of RWAs has been provided in [1.4].

[2] More details on the reason behind we use a different data sample will be given in 4.1.1.

rithms seen in the chapter [2.5].

4.1.1 The Data Samples

In order to assess the own risks and how they evolves over time the banks have to monitor their credit risk measures. Among those, the RWAs, cover a big chunk and every month this measure is analyzed.
In more detail, the delta between the current month end and the previous one is taken into account. Indeed, at the beginning of each month a monitoring report is created which analyzes the delta RWAs between 2 months end.
We want to understand which of these deltas are driven by business changes and which ones are anomalies that will give wrong information to the Risk Management influencing their decision making process.
Following this premise, we have created 2 data samples which are subsets of the data warehouse of worldwide company and they contain credit risk data where attributes such as RWAs, EAD, LGD, outstanding, etc. are selected. In other words, all credit risk attributes which could have a potential impact on the RWAs have been selected.
Nevertheless, in order to keep the process fast enough several filters have been applied on the row dimension of the dataset.
The reason behind the creation of 2 data samples is driven by the kind of analysis that will be performed. The first data sample will contain a monthly time series for around 142 distinct reporting dates[3]of 12631 distinct customers and it will be used for the time series analysis and data exploration which will help to understand the kind of phenomena that we want to study.
The second data sample is created for highlighting the data quality anomalies in the delta RWAs month end report.
The latter contains a much bigger amount of customers, 300848[4], but only 2 reporting dates.
These samples contains the same kind of information, we can imagine them as 2 different slices of the same cake which is represented by the data warehouse.

[3] From March 2007 to December 2018.

[4] In this step, it could be possible to get insights on the real portfolio composition of the company since not many filters have been applied. Hence, a randomization algorithm has been applied in order to mask the data. To safeguard the outcome of this thesis, it is ensured that at least 50000 rows are taken into account for the algorithm.

4.1.2 Extraction, cleaning and preparation of data

As seen in 4.1.1, we will use a specific data sample for the data exploration which is the starting step for all time series analysis, since it gives relevant information to understand what are the characteristics of the data that we are going to study.
During this exploration, most probably some transformations will be applied in order to create a workable dataset.
The extraction to create, clean and prepare the data sample is performed by SAS.
First of all, we have to understand how data looks like in the data warehouse and in our data sample, what kind of granularity and aggregation level has the sample and last but not least which slice of the data warehouse is going to be extracted for the data exploration step.
The Data Exploration step will mostly focus on the time series of the RWAs which is the phenomena that we want to study, so details about all other attributes which will be used in the modeling step are not analyzed. This analysis data extraction and exploration is performed by SAS Enterprise Guide.
As starting point, below are enumerated the filters that have been applied to the data warehouse for achieving the final data sample which is used for the data exploration step:

1. Select all active customers for all reporting dates available in the data warehouse;
2. Of which: customers treated under AIRB;
3. Of which: customers with customer type equal to 'Corporate';
4. Of which: customers within a single business unit;
5. Of which: customers have been part of the portfolio between 31^{st}March 2007 and 31^{st}December 2018;
6. Of which: customers with data available, in the selected dates, for at least 120 reporting dates out of the 142.

Selecting a particular customer type ('Corporate'), we exclude all records (customers) related to Governments, Individuals, Institutions, etc. This choice derives from the following reasons which in general apply to corporates:

1. They have much more dynamic business which will give some volatility to the RWAs. The continuous change of the RWAs has more probability to generate data anomalies;

2. Governments and Financial institutions run a less risky business than corporates. This is the reason why they have to be accurately monitored.

Besides, there is a filter applied on the Basel Approach. Indeed, rows where risk measures are calculated under the Standardized Approach (SA) have been discarded, in other words only records using internal methods for calculation (AIRB) have been selected.
A filter has been applied on a single business unit, this will reduce the number of customers in scope.
Last but not least, the latest 2 selection criteria in [4.1.2] have been applied in order to create a workable dataset, which exclude heavy data quality issues[5] and ensure enough data per each customer time series.
The only attributes considered for the data exploration are enumerated in table [4.1].

N.	Attributes
1	Reporting Date
2	Customer id
3	RWA

Table 4.1: Attributes selected from the data warehouse

The result of this filtering process is a dataset with 3 columns and 1.949.655 rows.
In the data warehouse, RWAs are available at the lowest level such as outstanding/cover.
In order to get more insights, we have aggregated[6] at customer id level which is the data level of our analysis.
The figure [10] shed more light on the granularity and structure of the data warehouse. Hence, the aggregation is done at the highest level, indeed, the sample goes from information at outstanding/cover level till to the customer level. In other words, this means a reduction of the number of rows. For example a customer which in the data warehouse has 3 rows per each reporting date, given his facility, outstanding and cover structure, it will appear

[5] Data Quality before March 2007 is considered to be really poor.
[6] less granular level

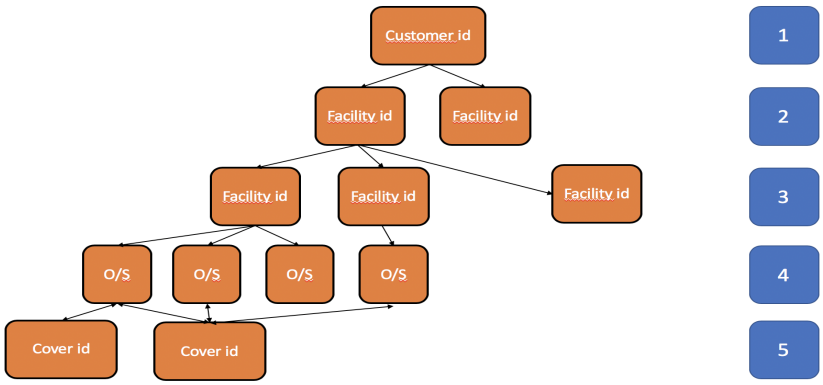

Figure 10: Granularity and structure of the data warehouse

as a single record since all information allocated over the structure will be aggregated at a higher level represented by the customer id [11] . Nevertheless, since we want to study the time-series of the RWAs of each customer, the RWAs information have been transposed and grouped per re- porting date.
The end result is a table with 142 rows, which represent the distinct reporting dates and 12631 columns where each column is a distinct customer id. In other words, each customer has a time series 142 months.

Customer	Facility	Outstanding	Cover	RWA
1	A	C	T	30
1	A	D	T	20
1	B	E	Z	10

Customer	RWA
1	60

Figure 11: Aggregation at customer level

Once, the data sample has been extracted and data has been put in the desired format we can start the data exploration.

4.1.3 Data Exploration

Visualize 12631 time series with a wide range of values is quite a challenge. The idea is to visualize few of them and run an ACF, PACF and IACF test on the same. Then, to generalize we automated the procedure and run the

the test on all customers in scope.
Once, the stationarity analysis has been completed and transformations have been applied, the Dickey-Fuller test will be performed again in order to assess that transformed series are stationary.
Hence, 3 random customers have been extracted from our dataset and plot the RWAs time series. The figure [12] will give a first idea of the behavior.

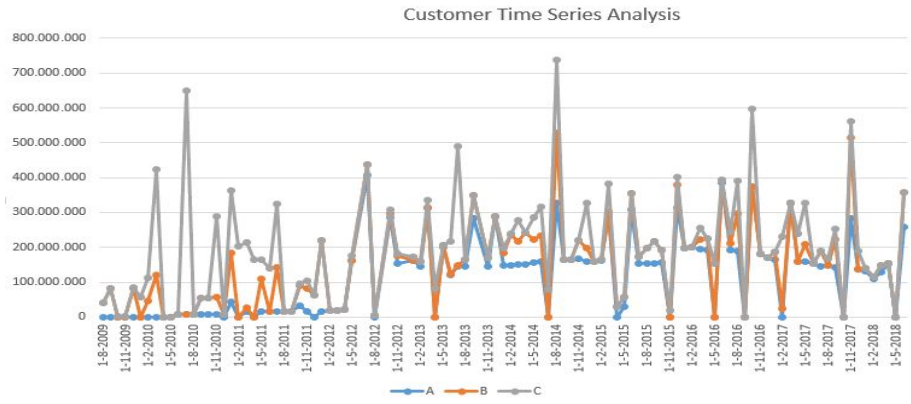

Figure 12: Customer Time Series Analysis

From a first look, all 3 RWAs time series seems to have a non-stationary behavior. The stationarity of the time series can be verified building the autocorrelation and partial autocorrelation functions. In order to proceed with the visualization we have selected the customer A and plotted those functions in the figure [13]. These time series are clearly non-stationary and this can strongly influence how to approach data in time series analysis as well explained in [56].
In more detail, the auto-correlation functions (ACFs) are, indeed, useful to evaluate the non-stationarity of the time series. In practice, for a stationary time series, we will observe that the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly.
The latter scenario is what we observe in the figure [13] which confirms the first hypothesis of a non stationary time series spotted in the figure [12].
To generalize, the same procedure is run on all customers. The table [4.2] emphasize that a big part of the data sample is not-stationary.

The customers in the table [4.2] who score with 0 are stationary while the ones which have 1 as result are non-stationary time series.
The Dickey-Fuller test do not accept the null hypothesis with α=0.05 when

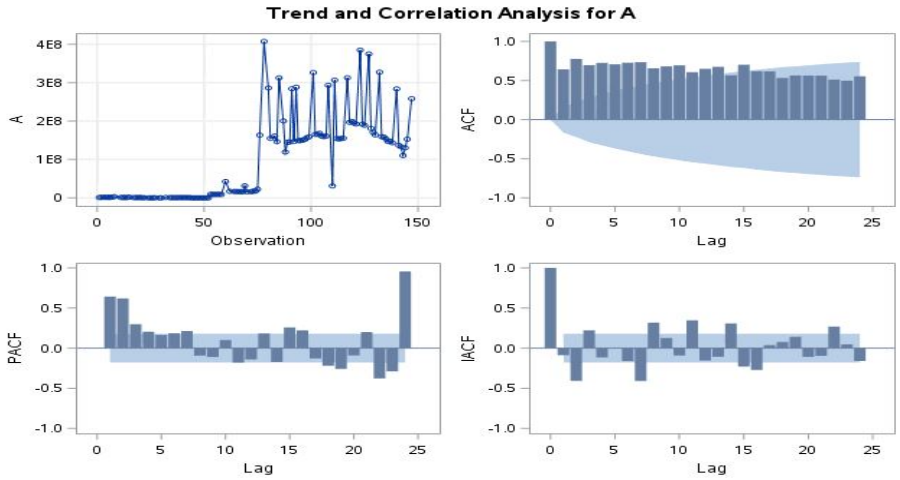

Figure 13: ACF, PACF and IACF of the RWAs time series of the customer A

Dickey-Fuller test	Number of Customers
0	16%
1	84%

Table 4.2: Dickey-Fuller Test

the p-value is less than α, which means that the time series are stationary. Hence, since we have concluded that part of our time series are non-stationary, we have to take some actions in order to transform the data which will allow us to apply most of the Statistical Techniques.
It is possible to divide this procedure in two steps:

1. Transform the time series, considering their logarithms;

2. Make the first difference of logarithms provided from the previous step.

Following this procedure, we can build again the ACF, PACF and IACF functions on the transformed data series. In order to visualize it, as first instance the random customer A is shown in the figure [14] and then it will be generalized to the entire data sample.
The figure [14] shows how the transformed time series have become stationary. In fact the spikes shown in the Figure [14] are common even in

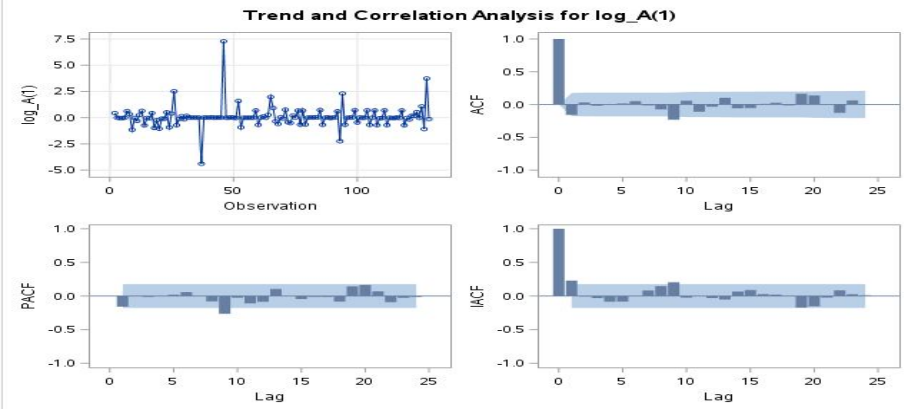

Figure 14: Customer A Time Series Analysis

stationary time series. In other words, after these two steps, the time series that can be analyzed with most of statistical techniques.
A further confirmation of the stationary series is provided from the plots of ACF in Figure [14]. It is evident that after first difference of logarithms the ACF drops relatively quickly drop to zero such as a cosine function; so these two transformations solve the non-stationarity problem.
The generalization of this procedure to the entire dataset brings to the results shown in table [4.3].

Dickey-Fuller test	Number of Customers
0	0%
1	100%

Table 4.3: Dickey-Fuller Test on the transformed RWA

After the log transformation and the first difference, all time series have become stationary.

4.2 Algorithm: Data, Calibration and Estimation

In 4.1.3, we have seen that in order to apply statistics techniques, transformation on the time series have to be applied.
Hence, to obtain stationary time series, data will be extracted, and the time

series will be transformed first with logarithms and then with the first difference.
As explained in 4.1.1, a specific data sample is used for the calibration of the algorithm and estimation of the data anomalies points.
Further details on how the sample is created, what data is selected and how it is aggregated are provided in [4.2.1].
The algorithm calibration and estimation steps do not focus on one single attribute as in [4.1.2], but they are related to most of the credit risk attributes which may influence the time series of the RWAs.
The main focus is on the isolation forest algorithm, the idea is to understand and verify, against reality, data anomalies and the capacity of the algorithm to detect those.
The extraction to create the data sample is performed by SAS Enterprise Guide as in [4.1.2], while the algorithm is run in Python.

4.2.1 Extraction, cleaning and preparation of data for algorithms application

As done in the subsection [4.1.2] for the data exploration part, it is practical to enumerate all steps done to derive the data sample. The steps below [4.2.1] show all filters applied to the data warehouse:

1. Select all active customers for 2 reporting dates, 30^{th}November 2018 and 31^{st}December 2018;

2. Of which: customers treated under AIRB;

3. Of which: customers with customer type equal to 'Corporate'.

The Table [4.4][7] gives a summary of the values in the data warehouse[8]and of the subset which will be utilized for this analysis. Hence, the data sample is a subset of the table [4.4] and the number of records part of it have been highlighted in bold.
This small summary table already gives some insights on the data sample and helps in pointing out some choices made in the data selection explained in [4.4].
One of the first considerations should be about the number of rows. From a quick sum, and keeping in mind that we have only 2 reporting dates in

[7]The real portfolio composition is masked using a randomization algorithm following the corporate data governance policy.
[8]For two reporting dates.

Date	Basel Appr	Cust Type	N.Cust	Delta Cust
30-Nov-2018	AIRB	Corporates	**300848**	
31-Dic-2018	AIRB	Corporates	**296411**	**-4437**
30-Nov-2018	AIRB	Governaments	6826	
31-Dic-2018	AIRB	Governaments	6953	127
30-Nov-2018	AIRB	Individuals	4195	
31-Dic-2018	AIRB	Individuals	4270	-75
30-Nov-2018	AIRB	Other Counterparties	9457	
31-Dic-2018	AIRB	Other Counterparties	9389	-68
30-Nov-2018	SA	Corporates	345399	
31-Dic-2018	SA	Corporates	341390	-4409
30-Nov-2018	SA	Governaments	700	
31-Dic-2018	SA	Governaments	700	0
30-Nov-2018	SA	Individuals	39	
31-Dic-2018	SA	Individuals	38	-1
30-Nov-2018	SA	Other Counterparties	256	
31-Dic-2018	SA	Other Counterparties	247	-9

Table 4.4: Summary of the Data Sample

scope, the dataset reaches over $1,3$Mio rows.
Besides, even if we have a small sample from the reporting dates point of view[9], it is already possible to see some changes in the portfolio composition[10] (e.g. number of corporates decreases of 4437, while the number of individuals increases of 127). This gives an idea of the volatility of the portfolio over time.
All the attributes selected are enumerated in table [4.5].
One of discussed problems in chapter [2.5] is related to the number of the attributes. Indeed, there are several algorithms (e.g. 2.2) which decrease their performances exponentially when the number of features increase.
In our subsample, we have taken into account 12 attributes[11] which have an impact on the RWAs calculation. These attributes should enable the algorithms to achieve the scope of the research, in other words to distinguish

[9] Potentially, data for more than 15 years on monthly basis at the lowest level is available. Besides, if the number of rows increase enormously, as understandable from [10], if a lower data level is considered, e.g. outstanding level.

[10] The real deltas in the portfolio composition are masked using a randomization algorithm following the corporate data governance policy.

[11] excluding the RWA

N.	Feature
1	Allocated Limit
2	Capital Requirement
3	EAD
4	LGD
5	Outstanding
6	PD
7	Provision
8	RW
9	RWA
10	Secured Recovery
11	Unsecured Amount
12	Unsecured Recovery
13	Unsecured Recovery Amount Discount

Table 4.5: Attributes selected form the data anomalies detection

between RWAs movements driven by business changes and the ones caused by anomalies. Most of these attributes have been covered with a brief explanation in Chapter [1.9].
Once the number of rows and columns have been selected, the next step is to perform the Isolation Forest algorithm.

4.2.2 Calibration of the Isolation Forest algorithm on Credit Risk Data

Once the data are filtered as explained in [4.2.1], 2 datasets related to November 2018 and December 2018 are exported from SAS Enterprise Guide and imported in Python.
This action helps since in Python, as well explained in [3], it is possible to call some libraries[12] which contain a pre-built Isolation Forest algorithm, where we have to set some parameters.
As done in [4.1.3], the series have been transformed.
Firstly, the logarithms of all relevant attributes have been calculated and secondly, the first differences have been applied.
In this case the first difference is equal to the delta between December and November.
In other words, this means that we will have a new dataset containing the

[12] sklearn.ensemble import IsolationForest

deltas of the logarithms of all variables and the Isolation Forest algorithm will be calibrated on top of this new dataset[13].
The Isolation Forest algorithm in Python requires some parameters in order to work.
The model has been defined as follow:

```
def init_model(parameters):
model = IsolationForest(max_samples=parameters['max_samples'],
                        max_features=parameters['max_features'],
                        n_estimators=parameters['n_estimators'],
                        n_jobs=-1,
                        contamination=parameters['contamination'])
return model
```

Listing 4.1: Model Definition

where max_samples, max_features, n_estimators, n_jobs and contamination have been already described in [3.2.2].
In this case the parameters have been set in the following way:

1. **max_samples** is equal to the 40% of the length of the dataset;
2. **max_features** has been set in a way that the algorithm pick 9 features of the 12 available;
3. **n_estimators** assumes a value of 64;
4. **n_jobs** has been set equal to -1 which means using all processors available [14];
5. **contamination** value is equal to 1.5%.

Besides for prediction of the results, we have used the following code:

```
def prediction(model, data, features):
            data['regular'] = model.predict(data[features])
            data['decision'] = model.decision_function(data[features])
            return data
```

Listing 4.2: Prediction

where the functions used, such as *predict* and *decision_function*, have been well explained in [3.2.2].
The entire code is available in the appendix [E].

[13] The dataset has been defined as *diff_data* in Python

[14] Parallel processing is helpful at improving runtime. This is usually a good idea to experiment rather than assuming that increasing the number of jobs is always a good thing

4.3 Discussion of the Results

In this paragraph, the main focus will be on the evidence of the results of the application of the Isolation Forest algorithm on our dataset.
The scope of the research is to spot the anomalies in the delta RWA data. Hence, as first step, we look at the distribution of the suspicious points which are the outcome of the IF algorithm.
Each observation of the output of the algorithm is labelled with 1 and -1 depending on how the observation are classified (regular vs. suspicious).
An insightful way to look at the distribution of the suspicious observation is using the violin plot showed in the figure [15].

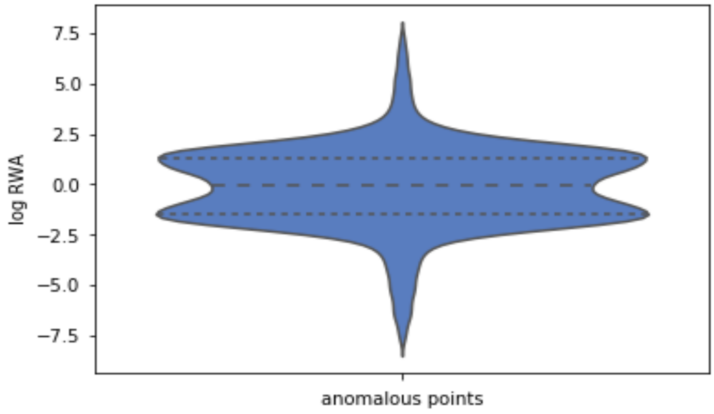

Figure 15: Log(RWA) distribution of the anomalous points

Violin plots are a valid alternative to box plots and although they are more informative they are still much less popular.
While a box plot only shows summary statistics such as median (in some cases even the mean is included) and interquartile range, the violin plot shows the full distribution of the data.
Indeed, in the figure [15], in addition to these summary statistics where the median of the data and the interquartile range are showed by dashed lines, the violin plot shows the distribution of quantitative data across several levels categorical variables such that those distributions can be compared.
Unlike a box plot, in which all of the plot components correspond to actual data points, the violin plot features a kernel density estimation of the underlying distribution.

The difference is particularly useful when the data distribution is multimodal, where the violin plot is able to show the presence of different peaks, their position and the relative amplitude.
Even if this can be an effective way to show multiple distributions at the same time, it is important to highlight that the estimation procedure is influenced by the sample size. In other words, violin plots for relatively small samples might look misleadingly smooth.
In this specific case, the graph [15] shows a distribution without any particular sign of asymmetry with a really high density close to zero.
These statements are confirmed from the descriptive statistics of the delta of the log RWA.
From the functional point of view, this can be read as cases where the delta RWA is close to zero even if other variables taken into consideration from the *IF* algorithm, such as LGD, EAD, RW are changing over time.
This kind of behavior of the delta RWA is not expected given the relation among the credit risk variables explained in [1.9] and is flagged as suspicious.

Index	delta log RRWA
mean	-0.028545
std	1.813199
min	-7.615207
25%	-1.396597
50%	0.000000
75%	1.392855
max	7.239503

Table 4.6: Delta LOG RRWA descriptive statistics

Once the distribution of the anomalous points is discussed is it valuable to focus on which points within the entire dataset the algorithm is recognizing as anomalies.
From the figure [16] is clearly visible how the algorithm is capable to flag as outliers not only the points that are in the tails, but also some points which are in the middle of the distribution.
The graph [16] shows the transformed data for the regulatory RWA.
The Python function used in this case is called *swarmplot*[15] where the

[15] This function is part of the *seaborn* package.

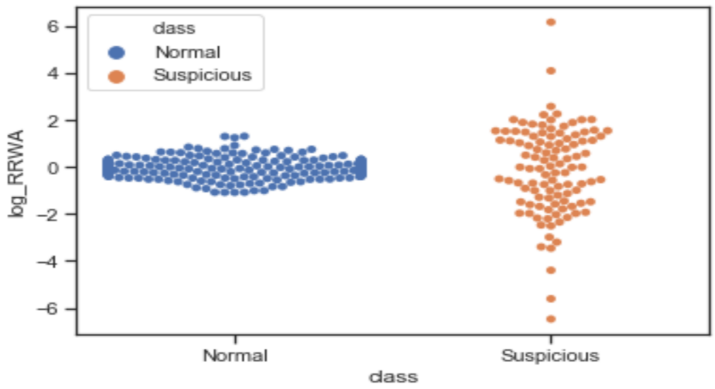

Figure 16: Anomalies vs Normal points of the log(RWA) time series

points are adjusted on categorical axis[16] so that they do notoverlap and create a distinction between classes.
In blue, we find the points which are considered inliers, while in orange the ones which have been classified as suspicious, or in other words, as a good candidates of being outliers.
The graph [16] confirms one of the strength of the IF algorithm which does not refer to a specific distribution as the density based algorithms showed in [2.2], so it is able to *isolate* the anomalies even if they are in a high density position.
This algorithm's property was not evident from our test on artificial samples in [3.2.3] since the anomalies have been randomly generated outside of the main clusters.
Hence, in the artificial example showed in [3.2.3], points detected as anomalies in the high density clusters, can be identified as false positive.
This has been the case for application of the LOF algorithm showed in [8], where some anomalies have been detected in the main clusters.
Last but not least, it is interesting to focus, while applying the IF on credit risk data, on the relation among the RWA and all other features taken into account.
Recalling the given relation between EAD and RWA showed in [1.4], RWA is direct function of the EAD and RW.
In other words, we are trying to underline that when RW and EAD are not changing, but we have a movement in RWA, most probably this will be a

[16] In our case, the x-axis shows 2 categories, Normal vs. Suspicious.

suspicious observation.
This assumption is confirmed by the figure [17] where it is clearly showed that Isolation Forest algorithm label most of the suspicious points[17] in the center (coordinate 0,0) where the delta of RW and EAD tend to zero.
Indeed, movements in RWA are not expected if the main drivers are not changing over time. In order to make this evidence, we have used *lmplot*[18]

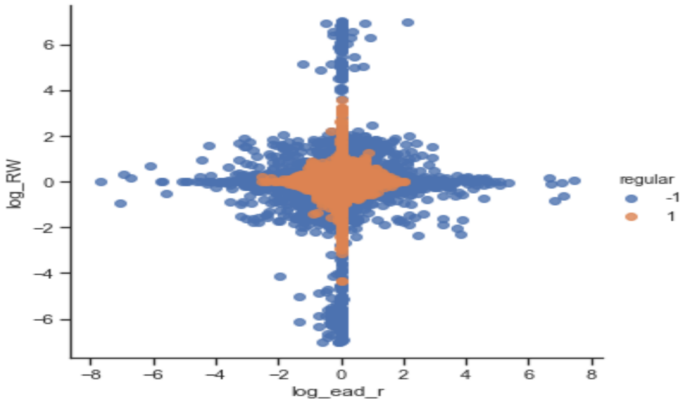

Figure 17: Anomalies vs Normal points of the log(RW) time series with respect to log(EAD)

which is a convenient interface to fit regression models across conditional subsets of a dataset.

4.4 Conclusions and next steps

One of the goal of this thesis was to show the application to credit risk data of outlier detection algorithms.
In particular, we have focused our attention on Isolation Forest algorithm and shown how his statistical properties are relevant in big data problems.
Indeed, we have showed that the algorithm is able to work in efficient way on huge datasets and it takes into account multiple dimensions or in other words it works with a number of features (k) much bigger than 2 which is

[17] In orange as in the graph [16].
[18] This function is part of the *seaborn* package.

in general the bottleneck for several outlier detection algorithms.
Besides, since the algorithm is not based on traditional statistical methods, it does not require assumptions on the underlying data distribution which from one side simplifies the model calibration while from the other allows the algorithm to label data as outliers independently from the position that they have within the data distribution (IF does not look only at the tails of the distribution).
Indeed, the delta RWAs identified as anomalies are not labelled as such because they are in the tails of the delta RWA distribution, but because these delta are not aligned with the movements of all other features taken into consideration from the IF algorithm for this analysis.
This aspect is clearly analyzed and observed in [4.3].
Nevertheless, this thesis is only a starting point of this analysis and much more can be achieved in order to show the goodness of this process.
In my opinion, the most relevant part, but also the most difficult to build and which require some time, is related to the backtest of the algorithm on the real credit risk dataset.
This would require the building of the information criteria as the ROC and PR curves described in [3.3].
In order to construct these curves TPR and FPR have to be available. This is possible only if we have in place a feedback loop mechanism on the algorithm output.
For testing our procedure, we are currently shortlisting [19] and sending the found anomalies to the local account managers underlining the reasons why the algorithm have recognized some specific observations for specific customers as data anomalies. In general, local account managers well know their portfolio and they are able to recognize and eventually amend some data anomalies. Indeed, in order to avoid that the same data anomaly will

[19] The IF algorithm used for this thesis requires a contamination parameter. In our case the contamination parameter has been set to $1,5\%$. This parameter describes the proportion of outliers expected in the data set. This small percentage in a big data problem can lead to a huge amount of data anomalies which are not manageable by the local account managers at the month end. Hence, a method to shortlist and prioritize the findings of the algorithm is strictly required. The program automatically sorts by descending delta RWA aggregated at customer level and send the top ten customers to the local account managers. The focus of the entire exercise is on delta RWA since this is one of the most relevant risk measures for all banks which attracts a lot of attention by bank regulators and supervisors. Given the scope of this project, that introduces and tests the reliability of data anomalies algorithms in a credit risk environment, we have decided to limit to a maximum of 10 customers the data anomalies sent to each local account manager.

pop up again in the next reporting period[20], the local account manager is asked to further investigate the source of the data anomaly and take actions if needed.
Most of the times, we have received a positive feedback on the findings, which is an additional reason to believe that the IF is a great method to detect data anomalies and this does not apply only on artificial samples as showed in [3.2.3] but on real credit risk dataset as well.
The extra step to achieve, in order to have enough data for the construction of the information criteria, is to build a database on the received feedback. This will have two big benefits for our model:

1. The algorithm will receive a feedback on the results and can learn from it. This will help in model calibration;

2. We can build the ROC and the PR curves.

In other words, this extra steps will give a much more clear idea on the performance reached by the algorithm and it will help it in future predictions.

[20] We want to avoid an incorrect population of the monthly reports. Hence, the idea is to run this process once a month. Alternative scenario is when we want to test the correct population of the data warehouse which is the source of the monthly reports. With respect to this scenario, we have to consider that the central data warehouse, on which we are running the algorithm, is populated daily for retail data and weekly for business. Since the scope is reduced to the customers with high RWA, we can assume the retail customers will be excluded a priori. In oder words, in this scenario the most indicated frequency to run the IF algorithm is weekly.

Appendix

Appendix A

Nested-Loop Algorithm for Density-Based Approach

Algorithm NL

1. Fill the first array (of size $\frac{B}{2}\%$ of the dataset) with a block of tuples from T.

2. For each tuple ti in the first array, do:

   a. $count_i \leftarrow 0$

   b. For each tuple t_j in the first array, if dist(t_i, t_j)$\leq D$:
   Increment $count_i$ by 1. If $count_i > M$, mark t_i as a non-outlier and proceed to next t_i.

3. While blocks remain to be compared to the first array, do:

   a. Fill the second array with another block (but save a block which has never served as the first array, for last).

   b. For each unmarked tuple t_i in the first array do:
   For each tuple t_j in the second array, if $dist(t_i, t_j) \leq D$: Increment $count_i$ by 1. If $count_i > M$, mark t_i as a non-outlier and proceed to next t_i.

4. For each unmarked tuple t_i in the first array, report t_i as an outlier.

5. If the second array has served as the first array anytime before, stop; otherwise, swap the names of the first and second arrays and goto step 2.

Appendix B

Squeezer algorithm

Let $A_1, \ldots, A_m$ be a set of categorical attributes with domains $D_1, \ldots, D_m$ respectively. Let the dataset D be a set of tuples where each tuple $t : t \in D_1 \times \cdots \times D_m$. Let TID be the set of unique identifier of every tuple. For each $tid \in TID$, the attribute value for A_i of corresponding tuple is represented as $tid \cdot A_i$.
Starting from the concept that a cluster is a subset of TID, then the following definitions are important for building the Squeezer algorithm:

Definition 21 *Given a cluster C, the set of different attribute value on A_i with respect to C is defined as:*

$$VAL_i(C) = \{tid \cdot A_i | tid \in C\} \tag{B.1}$$

where $1 \geq i \geq m$.

Definition 22 *Given a Cluster C, let $a_i \in D_i$, the support of a_i in C with respect to A_i is defined as:*

$$Sup(a_i) = |\{tid | tid | A_i = a_i, tid \in C\}| \tag{B.2}$$

Definition 23 *Given a Cluster C, the Summary for C is defined as:*

$$Summary = \{VS_i | 1 \geq i \geq\} where VS_i = \{(a_i, Sup(a_i)) | a_i \in VAL_i(C)\}. \tag{B.3}$$

In other words, the summary contains m elements, where m is number of attributes, and gives information about the cluster.

Definition 24 *Given a cluster C, the cluster structure (CS) for C is defined as:*

$$CS = \{cluster, summary\}. \tag{B.4}$$

Definition 25 *Given a cluster C and a tuple t with $tid \in TID$, the similarity between C and tid is defined as:*

$$Sim(C, tid) = \sum_{i=1}^{m} \left(\frac{Sup(a_i)}{\sum_{a_j \in VAL_i(C)} Sup(a_j)} \right) \tag{B.5}$$

where $a_i = tid \cdot A_i$

In the Squeezer algorithm, the similarity as defined in B.5 is used to determine whether the tuple should be put into the cluster or not. The Squeezer algorithm has n tuples as input and produce clusters as final results.
For every tuple, by the $similarityfunction$, the similarity is computed with all existing clusters and the largest value of $similarity$ is selected out. If it is larger than the given threshold, the tuple will be inserted into the cluster that has the largest value of $similarity$ and the CS will be updated. If the above condition does not hold, a new cluster is created with this tuple. The algorithm continues until it has traversed all the tuples in the dataset.
As in [45] the Squeezer algorithm can be summarized as follow: The first

	Algorithm Squeezer (D,s)
1	**Begin**
2	**while** (D has unread tuple){
3	tuple=getCurrentTuple (D)
4	**if** (tuple.tid ==1) {
5	addNewClusterStructure (tuple.tid)}
6	**else** {
7	for each existed cluster C
8	simComputation (C, tuple)
9	get the max value of similarity: sim_max
10	get the corresponding Cluster Index: index
11	**if** $sim_max \geq$s
12	addTupleToCluster (tuple, index)
13	**else**
14	addNewClusterStructure (tuple.tid)}
15	}
16	outputClusteringResult()
17	**End**

Table B.1: Squeezer algorithm

tuple is read in, and the sub-function $addNewClusterStructure()$ is used

to establish a new CS, which includes summary and cluster (Steps 3-4).
Then the similarity between an existed cluster C and each tuple is computed using sub-function $simComputation()$. Once the maximal value of similarity is taken(sim_max) and the corresponding index of cluster ($index$)(Steps 6-9).
If the max of the similarity is larger than the input threshold s, then the sub-function $addTupleToCluster()$ insert the tuple to selected cluster (steps 10-11).
If the previous condition does not hold, the sub-function $addNewClusterStructure()$ constructs a new CS (Steps 12-13).
As final step, the clustering results are labeled on the disk (Step 15).

The Squeezer algorithm is a background clustering algorithm for outlier detection which own the following characteristics:

a. Achieves both high quality of clustering results and scalability;

b. Ability in handling effectively high dimensional data-sets;

c. Does not require the number of desired clusters as an input parameter and it can produce more natural clusters with significant different sizes.

Appendix C

FindCBLOF algorithm

Algorithm *Find**CBLOF***

Input: $D(A_1, ..., A_m)$, // the data set
The parameter α and β // parameters

Output: *The values of CBLOF for all records* //indicates the degree of deviation

```
01 begin
02      Clustering the dataset D(A1, ..., Am) using Squeezer algorithm
03      /* Produced clusters: C = {C1, C2, ..., Ck} and |C1|>=|C2|>= ...>=|Ck| */
04      Get LC and SC with the 2 parameters   //get the set of large and small clusters
05      foreach record t in the dataset do begin
06          if t belongs to Ci and Ci belongs to SC do begin
07              CBLOF= |Ci|*min (distance (t, Cj))     // Cj belongs to LC
08          else
09              CBLOF = |Ci|*distance(t, Ci)          // Ci belongs to LC
10          return CBLOF
11      end
12  end
```

Figure 18: CBLOF algorithm

Appendix D

Path-Length-Based Isolation

Following [52], the iForest can be implemented with the following 3 Algorithms:

1. Algorithm for the training stage [D.1];

	Algorithm 1: iForest (X, t, ψ)
1	**Inputs**: X - input data, t - number of trees, ψ - subsampling size
2	**Output**: a set of t iTrees
3	**Initialize**: Forest
4	**for** i=1 to t do
5	$X' \leftarrow$ sample (X,ψ)
6	Forest $\leftarrow$ Forest $\cup$ iTree (X')
7	**end for**
8	**return** Forest

Table D.1: iForest and training stage

2. Training and evaluation stage of a single path length $h(x)$ [D.2];

3. The path length algorithm gives additional details of the evaluation stage [D.3].

	Algorithm 2: iTree(X')
1	**Inputs**: X' - input data
2	**Output**: an iTree
3	**if** X' cannot be divided **then**
4	return exNode { Size $\leftarrow |X'|$ }
5	**else**
6	let Q be a list of attributes in X'
7	randomly select an attribute $q \in Q$
8	randomly select a split point p between the max and the min values of attribute
9	q in X'
10	$X_l \leftarrow$ filter(X',$q < p$)
11	$X_r \leftarrow$ filter(X',$q \geq p$)
12	return inNode {Left $\leftarrow$ iTree (X_l),
13	Right $\leftarrow$ iTree (X_r),
14	SplitAtt $\leftarrow$ q,
15	SplitValue $\leftarrow$ p}
16	**end if**

Table D.2: iTree and training stage

	Algorithm 3: PathLenght(x, T, $hlim$, e)
1	**Inputs**: x - an instance, T - an iTree, $hlim$ - height limit, e - current path length;
2	to be initialized to zero when first called
3	**Output**: path length of x
4	**if** T is an external node or $e \geq$hlim **then**
5	return e+c(T.size){$c(.)$ is defined in Equation 1}
6	**end if**
7	a $\leftarrow$ T.splitAtt
8	**if** $x_a < T$.splitValue **then**
9	return PathLength (x, T.left, $hlim$, $e+1$)
10	**else** {$x_a \geq T$.splitValue}
11	return PathLength (x, T.right, $hlim$, $e+1$)
12	**end if**

Table D.3: Path Length

Appendix E

Python Procedures

```
import pandas as pd
import numpy as np
import io
import base64
import random
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
from sklearn.model_selection import train_test_split
import time
from sklearn import svm
from sklearn.datasets import make_moons, make_blobs
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

from itertools import cycle
from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp

def read_file(filename):
    file_extension = filename.split(".")[-1]
    if file_extension in ['xls', 'xlsx']:
        excel_file = pd.ExcelFile(filename)
        data = excel_file.parse(excel_file.sheet_names[0])
    elif file_extension in ['csv', 'txt']:
        data = pd.read_csv(filename)
    else:
        raise (IOError('File type not supported'))

    return data
```

```
38
39 plt.rcParams['contour.negative_linestyle'] = 'solid'
40
41 # Example settings
42 n_samples = 30000
43 outliers_fraction = 0.10
44 n_outliers = int(outliers_fraction * n_samples)
45 n_inliers = n_samples - n_outliers
46
47 # define outlier/anomaly detection methods to be compared
48 anomaly_algorithms = [
49     ("Isolation Forest", IsolationForest(behaviour='new',
50                                          contamination=outliers_
                                                 fraction,
51                                          random_state=42)),
52     ("Local Outlier Factor", LocalOutlierFactor(
53         n_neighbors=35, contamination=outliers_fraction))]
54
55 # Define datasets
56 blobs_params = dict(random_state=0, n_samples=n_inliers, n_features=2)
57 datasets = [
58     make_blobs(centers=[[0, 0], [0, 0]], cluster_std=0.5,
59                **blobs_params)[0],
60     make_blobs(centers=[[2, 2], [-2, -2]], cluster_std=[0.5, 0.5],
61                **blobs_params)[0],
62     make_blobs(centers=[[2, 2], [-2, -2]], cluster_std=[1.5, .3],
63                **blobs_params)[0]]
64
65 # Compare given classifiers under given settings
66 xx, yy = np.meshgrid(np.linspace(-7, 7, 150),
67                      np.linspace(-7, 7, 150))
68
69 plt.figure(figsize=(len(anomaly_algorithms) * 2 + 3, 12.5))
70 plt.subplots_adjust(left=.02, right=.98, bottom=.001, top=.96, wspace
       =.05,
71                     hspace=.01)
72
73 plot_num = 1
74 rng = np.random.RandomState(42)
75
76 for i_dataset, X in enumerate(datasets):
77     # Add outliers
78     X = np.concatenate([X, rng.uniform(low=-6, high=6,
79                         size=(n_outliers, 2))], axis=0)
80
81     for name, algorithm in anomaly_algorithms:
82         t0 = time.time()
83         algorithm.fit(X)
84         t1 = time.time()
85         plt.subplot(len(datasets), len(anomaly_algorithms), plot_num)
86         if i_dataset == 0:
87             plt.title(name, size=18)
88
89         # fit the data and tag outliers
90         if name == "Local Outlier Factor":
91             y_pred = algorithm.fit_predict(X)
```

```
92          else:
93              y_pred = algorithm.fit(X).predict(X)
94
95          # plot the levels lines and the points
96          if name != "Local Outlier Factor":  # LOF does not implement
                predict
97              Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
98              Z = Z.reshape(xx.shape)
99              plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='
                    black')
100
101         colors = np.array(['#377eb8', '#ff7f00'])
102         plt.scatter(X[:, 0], X[:, 1], s=10, color=colors[(y_pred + 1)
                // 2])
103
104         plt.xlim(-7, 7)
105         plt.ylim(-7, 7)
106         plt.xticks(())
107         plt.yticks(())
108         plt.text(.99, .01, ('%.2fs' % (t1 - t0)).lstrip('0'),
109                  transform=plt.gca().transAxes, size=15,
110                  horizontalalignment='right')
111         plot_num += 1
112
113 plt.show()
114
115 features_code = ['EAD', 'Provisions',
116                  'Outsatnding', 'RWA', 'RW', 'LGD', 'PD',
117                  'Allocation Limit', 'Capital Requirement',
118                  'Secured Recovery', 'Unsecured Recovery',
119                  'Unsecured Amount', 'Unsecured Recovery Amount
                         Discount']
120
121
122 def prepare_data(model_dic):
123     print('CAL preparing data -')
124     data_pre = model_dic['previous'][(model_dic['previous']['customer_
            type_lvl1']=='Corporates') &
125                      (model_dic['previous']['approach']=='AIRB')]
126     data_cur = model_dic['current'][(model_dic['current']['customer_
            type_lvl1']=='Corporates') &
127                      (model_dic['current']['approach']=='AIRB')]
128
129     features = ['ead_r', 'provisions_r',
130             'OS_r', 'RRWA', 'RW', 'lgd_r_wa', 'pd_r_wa',
131             'alloc_limit', 'capital_requirement',
132             'secured_recovery_amt_disc',
133             'unsecured_amount', 'unsecured_recovery_amts_di']
134
135     agg_cols = ['customer_id', 'committed_ind', 'product_type_lvl1_
            descr']
136     ead_col = find_ead_col(data_pre.columns, features)
137     if ead_col == '':
138         raise ValueError('EAD columns is not available.')
139
140     prec_cols = split_prec_scal(data_pre, agg_cols)
```

```
141
142     print('CAL aggregating data -')
143     data_pre = aggregate_data(data=data_pre, agg_cols=agg_cols,
144                               ead_col=ead_col, prec_cols=prec_cols)
145     data_cur = aggregate_data(data=data_cur, agg_cols=agg_cols,
146                               ead_col=ead_col, prec_cols=prec_cols)
147
148     model_dic['data_pre'] = data_pre
149     model_dic['data_cur'] = data_cur
150
151     print('CAL adding log columns -')
152     data_pre = add_log_cols(data=data_pre, features=features, prec_
            cols=prec_cols)
153     data_cur = add_log_cols(data=data_cur, features=features, prec_
            cols=prec_cols)
154
155     model_dic['log_cols'] = ['log_' + col for col in features]
156     print('CAL creating diff data -')
157     diff_data = data_cur[model_dic['log_cols']] - data_pre[model_dic['
            log_cols']]
158     diff_data = diff_data.dropna()
159     model_dic['diff_data'] = diff_data
160
161     return model_dic['data_pre'], model_dic['data_cur'], model_dic['
            diff_data'], model_dic['log_cols']
162 def reading_data(data_pre, data_cur):
163     data_dic = {}
164     data_dic['previous'] = data_pre
165     data_dic['current'] = data_cur
166     return data_dic
167 def generate_diff_data(data_dic):
168     data = data_dic['current'] - data_dic['previous']
169     data = data.dropna()
170     return data
171 def select_scl_cols(data, cols):
172     scl_cols = []
173     prc_cols = []
174
175     for col in cols:
176         if data[col].max() > 1.0:
177             scl_cols.append(col)
178         else:
179             prc_cols.append(col)
180
181     return scl_cols, prc_cols
182
183 def find_ead_col(columns, features):
184     ead_cols = []
185     for item in features:
186         if 'ead' in item:
187             ead_cols.append(item)
188
189     if len(ead_cols) == 1:
190         return ead_cols[0]
191
192     if len(ead_cols) > 1:
```

```
193         for item in ead_cols:
194             if 'ead_r' in item:
195                 return item
196         return ead_cols[0]
197 
198     ead_cols = []
199     for item in columns:
200         if 'ead' in item:
201             ead_cols.append(item)
202     if len(ead_cols) == 1:
203         return ead_cols[0]
204     elif len(ead_cols) > 1:
205         for item in ead_cols:
206             if 'ead_r' in item:
207                 return item
208         return ead_cols[0]
209 
210 
211 def split_prec_scal(data, ignore):
212     prec_cols = []
213     numeric_cols = data._get_numeric_data().columns.get_values()
214    non_index_numeric_cols = [x for x in numeric_cols if x not in
           ignore]
215     for ftr in non_index_numeric_cols:
216         if data[ftr].max() < 100.0:  # RW could be larger than 1
               although it has to be weighted average by EAD
217             prec_cols.append(ftr)
218 
219     return prec_cols
220 
221 
222 def add_log_cols(data, features, prec_cols):
223     for col in features:
224         if col in prec_cols:
225             data['log_' + col] = np.log10(data[col] + 1.0e-6)
226         else:
227             data['log_' + col] = np.log10(data[col] + 1.0)
228     return data
229 
230 
231 def aggregation_function(data, non_numeric):
232     return data[non_numeric][0:1]
233 
234 
235 def aggregate_data(data, agg_cols, ead_col, prec_cols):
236     data.index.name = 'default_index'
237     new_cols = []
238     for col in prec_cols:
239         new_col_str = ead_col + '*' + col
240         new_cols.append(new_col_str)
241         data[new_col_str] = data[ead_col] * data[col]
242 
243     agg_cols = [x for x in agg_cols if x != '']
244     print('CAL grouping by data -')
245     numeric_cols = data._get_numeric_data().columns.get_values()
246     non_numeric = [x for x in data.columns if x not in numeric_cols]
```

```
247
248     df_num = data.groupby(agg_cols)[numeric_cols].sum()
249     for col in prec_cols:
250         new_col_str = ead_col + '*' + col
251         df_num[col] = df_num[new_col_str]/df_num[ead_col]
252
253     df_num = df_num.drop(new_cols, axis=1)
254
255     df_non = data.groupby(agg_cols).apply(aggregation_function, non_
            numeric)
256     for col in agg_cols:
257         if col in df_non.columns:
258             df_non = df_non.drop(col, axis=1)
259
260     df_non = df_non.reset_index().set_index(agg_cols)
261     df_non = df_non.drop('default_index', axis=1)
262
263     df_final = df_non.join(df_num, how='left')
264
265     return df_final
266
267
268 def make_filter(model_dic):
269     filters = {}
270     for flt in model_dic['filters_cols']:
271         filters[flt] = model_dic[flt]
272
273     return filters
274
275 def filter_data(data, filters):
276     data_tmp = data
277     for flt in filters.keys():
278         data_tmp = data_tmp[data_tmp[flt] == filters[flt]]
279
280     return data_tmp
281 def calibrate_model(data, cont_factor, features):
282     parameters = {}
283     parameters['max_samples'] = 0.4
284     if len(features) > 12:
285         parameters['max_features'] = 9
286     else:
287         parameters['max_features'] = len(features)
288     parameters['n_estimators'] = 64
289     parameters['n_jobs'] = -1
290     parameters['contamination'] = cont_factor
291
292     print('CAL initializing model -')
293     model = init_model(parameters=parameters)
294
295     print('CAL fitting model -')
296     model = fit_model(model=model,
297                       data=data,
298                       features=features)
299
300     print('CAL scoring data -')
301     data = prediction(model=model,
```

```
                        data=data,
                        features=features)

    return data, model


def apply_model(model, data, features):
    print('CAL scoring data -')
    data = dm.prediction(model=model,
                        data=data,
                        features=features)

    return data

def init_model(parameters):
    model = IsolationForest(max_samples=parameters['max_samples'],
                            max_features=parameters['max_features'],
                            n_estimators=parameters['n_estimators'],
                            n_jobs=-1,
                            contamination=parameters['contamination'])

    return model


def fit_model(model, data, features):
    model.fit(data[features])

    return model


def prediction(model, data, features):
    data['regular'] = model.predict(data[features])
    data['decision'] = model.decision_function(data[features])

    return data

data_cur = read_file('FINAL_DATA_RECENT.xlsx')
data_pre = read_file('FINAL_DATA_PREVIOUS.xlsx')

model_dic = reading_data(data_pre,data_cur)

model_dic['data_pre'], model_dic['data_cur'],model_dic['diff_data'],
    model_dic['log_cols'] = prepare_data(model_dic)

model_dic['diff_data'], model_dic['model'] =
calibrate_model(data=model_dic['diff_data'],
                           cont_factor=0.015,
                          features=model_dic['log_cols'])
model_dic['diff_data'].sort_values(
            'decision', ascending=True)

model_dic['diff_data'][model_dic['diff_data']['regular']==-1].count()/
    model_dic['diff_data'].count()


```

```
355 _ = sns.violinplot(y="log_RRWA", hue="regular",data=model_dic['diff_
        data'][model_dic['diff_data']['regular']==-1], palette="muted",
        split=True,scale="count", inner="quartile")
356 _ = plt.xlabel('anomalous points')
357 _ = plt.ylabel('log RWA')
358 plt.show()
359
360 sns.set(style="ticks")
361
362 model_dic['diff_data']['log_RRWA'][model_dic['diff_data']['regular'
        ]==-1].describe()
363
364 for i in range (0,2):
365     df_test = {}
366     df_test = pd.DataFrame(model_dic['diff_data'][model_dic['diff_data
            ']['regular']!=-1].sample(frac=0.0033, replace=False,random_
            state=i).append(
367     model_dic['diff_data'][model_dic['diff_data']['regular']==-1].
            sample(frac=0.02167, replace=False,random_state=i)))
368
369     df_test['class'] = np.where(df_test.regular == 1, 'Normal',
370                                 'Suspicious')
371
372
373     sns.swarmplot(x="class", y ='log_RRWA', data = df_test,hue = '
            class')
374
375     plt.show()
376
377 # fit the model for outlier detection (default)
378 clf = LocalOutlierFactor(n_neighbors=20, contamination=0.015)
379 # use fit_predict to compute the predicted labels of the training
        samples
380 # when LOF is used for outlier detection, the estimator has no predict
        , decision_function and score_samples methods).
381 y_pred = clf.fit_predict(np.array(data_train[['log_RRWA','log_ead_r'
        ]]))
382 X_scores = clf.negative_outlier_factor_
383 X=(np.array(data_train[['log_RRWA','log_ead_r']]))
384
385 plt.title("Local Outlier Factor (LOF)")
386 plt.scatter(X[:, 0], X[:, 1], color='k', s=3., label='Data points')
387 # plot circles with radius proportional to the outlier scores
388 radius = (X_scores.max() - X_scores) / (X_scores.max() - X_scores.min
        ())
389 plt.scatter(X[:, 0], X[:, 1], s=2000 * radius, edgecolors='r',
390             facecolors='none', label='Outlier scores')
391 plt.axis('tight')
392 plt.xlim((-0.01, 0.01))
393 plt.ylim((-0.01, 0.01))
394 legend = plt.legend(loc='upper left')
395 legend.legendHandles[0]._sizes = [10]
396 legend.legendHandles[1]._sizes = [20]
397 plt.show()
```

Listing E.1: Python Procedures

References

[1] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509. ACM, 2006. iv, 40, 45, 46, 47, 48

[2] Edward I Altman, Brooks Brady, Andrea Resti, and Andrea Sironi. The link between default and recovery rates: Theory, empirical evidence, and implications. *The Journal of Business*, 78(6):2203–2228, 2005. 17

[3] Vic Barnett and Toby Lewis. *Outliers in statistical data*. Wiley, 1974. 37, 38

[4] II Basel. Revised international capital framework. *Basel, Switzerland: Basel Committee on Banking Supervision*, 2004. 8, 11, 14

[5] BCBS Basel III. The liquidity coverage ratio and liquidity risk monitoring tools. *Bank for International Settlements*, 2013. 5

[6] Stefan Benvegnù, Christian Bluhm, and Christoph Müller. *A guide to active credit portfolio management: spotlight on illiquid credit risks*. Risk Book, 2008. 23

[7] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016. 49

[8] B BIS. Basel iii: International regulatory framework for banks, 2011. 9, 11

[9] Christian Bluhm and Ludger Overbeck. *Structured credit portfolio analysis, baskets and CDOs*. Chapman and Hall/CRC, 2006. 20

[10] Christian Bluhm, Ludger Overbeck, and Christoph Wagner. *An introduction to credit risk modeling*. Chapman and Hall/CRC, 2002. 19, 27

[11] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997. 76

[12] Friedman JH Olshen RA Breiman, Leo and CJ Stone. *Classification and regression trees*. Routledge, 2017. 36, 40, 42, 45

[13] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 45, 48

[14] Leo Breiman. Out-of-bag estimation. Technical report, Department of Statistics, University of Berkeley (U.S.A.), 1996. 51

[15] Leo Breiman. Using adaptive bagging to debias regressions. Technical report, Technical Report 547, Statistics Dept. UCB, 1999. 48

[16] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 40, 45, 48, 50, 51, 65

[17] Leo Breiman, JH Friedman, RA Olshen, and CJ Stone. Classification and regression trees. wadsworth & brooks. *Cole Statistics/Probability Series*, page 2, 1984. 41

[18] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000. 36, 46, 48, 52, 58, 64

[19] Markus K Brunnermeier and Lasse Heje Pedersen. Market liquidity and funding liquidity. *The review of financial studies*, 22(6):2201–2238, 2008. 5

[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009. 33

[21] Michel Crouhy, Dan Galai, and Robert Mark. A comparative analysis of current credit risk models. *Journal of Banking & Finance*, 24(1-2):59–117, 2000. 20

[22] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006. 73

[23] Comitato di Basilea per la vigilanza bancaria. *International convergence of capital measurement and capital standards: a revised framework*. Bank for International Settlements, 2004. 26

[24] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000. 45, 48, 50

[25] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2018. 33, 46, 63, 77

[26] David L Donoho, Miriam Gasko, et al. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4):1803–1827, 1992. 38

[27] Mathias Drehmann and Kleopatra Nikolaou. Funding liquidity risk: definition and measurement. *Journal of Banking & Finance*, 37(7):2173–2182, 2013. 5

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 32, 52

[29] Eric Falkenstein, Andrew Boral, and Lea Carty. Riskcalc for private companies: Moody's default model. 2000. 22

[30] Ramsey Faragher et al. Understanding the basis of the kalman filter via a simple and intuitive derivation. *IEEE Signal processing magazine*, 29(5):128–132, 2012. 33

[31] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006. 76

[32] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996. 32

[33] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951. 44

[34] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999. 46

[35] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 46, 48

[36] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer, 1996. 46

[37] Sebastian Fritz, Michael Luxenburger, and Thomas Miehe. *Implementation of an IRB compliant rating system*. Risk Books, London, 2004. 21

[38] Ram Gnanadesikan, Jon R Kettenring, and James M Landwehr. Interpreting and assessing the results of cluster analyses. *Bulletin of the International Statistical Institute*, 47(2):451–463, 1977. 44

[39] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE, 1999. 52

[40] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *VLDB*, volume 18, pages 574–559, 1992. 32

[41] David J Hand. Kernel discriminant analysis. *JOHN WILEY & SONS, INC., ONE WILEY DR., SOMERSET, N. J. 08873, 1982, 264*, 1982. 44

[42] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001. 76

[43] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. 76

[44] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980. 33, 37, 38, 52, 64

[45] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003. 36, 52, 53, 54, 100

[46] Bengt Holmström and Jean Tirole. Lapm: A liquidity-based asset pricing model. *the Journal of Finance*, 56(5):1837–1867, 2001. 5

[47] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002. 46

[48] John Maynard Keynes. *A treatise on money: in 2 volumes*. Macmillan & Company, 1930. 5

[49] 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. 32, 33, 38, 39, 52, 55

[50] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166. ACM, 2005. 48

[51] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. 36, 54, 56, 57, 63

[52] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):3, 2012. 36, 54, 56, 57, 58, 59, 60, 63, 103

[53] Naoki Abe Hiroshi Mamitsuka et al. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML?98)*, volume 1. Morgan Kaufmann Pub, 1998. 46

[54] Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *KDD*, volume 96, pages 146–151, 1996. 32

[55] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovering frequent episodes in sequences extended abstract. In *1st Conference on Knowledge Discovery and Data Mining*, 1995. 32

[56] F.Giordano; M.Niglio and Cosimo D.Vitale. *Processi stocastici ed inferenza statistica*. Edizioni Scientifiche Italiane, 2014. 84

[57] Ray Bradford Murphy. *On tests for outlying observations*. PhD thesis, Princeton University, 1951. 55

[58] Alexandros Nanopoulos, Yannis Theodoridis, and Yannis Manolopoulos. C2p: clustering based on closest pairs. In *VLDB*, pages 331–340, 2001. 52

[59] Kleopatra Nikolaou. Liquidity (risk) concepts: definitions and interactions. 2009. 4

[60] Bruno R Preiss. *Data structures and algorithms with object-oriented design patterns in C++*. John Wiley & Sons, 2008. 59

[61] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012. 38

[62] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29, pages 427–438. ACM, 2000. 33, 52

[63] Riccardo Rebonato. Theory and practice of model risk management. *Modern Risk Management: A History?, RiskWaters Group, London*, pages 223–248, 2002. 7

[64] Ida Ruts and Peter J Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, 23(1):153–168, 1996. 38, 39

[65] Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998. 46

[66] William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964. 2, 3

[67] Kent A Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163. Elsevier, 1989. 74

[68] Philip Strahan. Liquidity production in 21st century banking. Technical report, National Bureau of Economic Research, 2008. 5

[69] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to data mining. 1st, 2005. 37, 38

[70] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto(Canada), 1996. 51

[71] John W Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, pages 523–531, 1975. 38

[72] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 37

[73] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999. 37

[74] Vladimir N Vapnik and A Ja Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk. *Pattern Recognition and Image Analysis*, 1(3):284–305, 1991. 37

[75] Tom Wilde and Lee Jackson. Credit portfolio risk low-default portfolios without simulation. *RISK-LONDON-RISK MAGAZINE LIMITED-*, 19(8):60, 2006. 23

[76] Cristiano Zazzara. Il ruolo del capitale nelle banche e la sua regolamentazione: dall?accordo di basilea del 1988 ad oggi. *Rivista Minerva Bancaria*, (5), 1999. 8

[77] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996. 32

Acknowledgements

acknowledgements