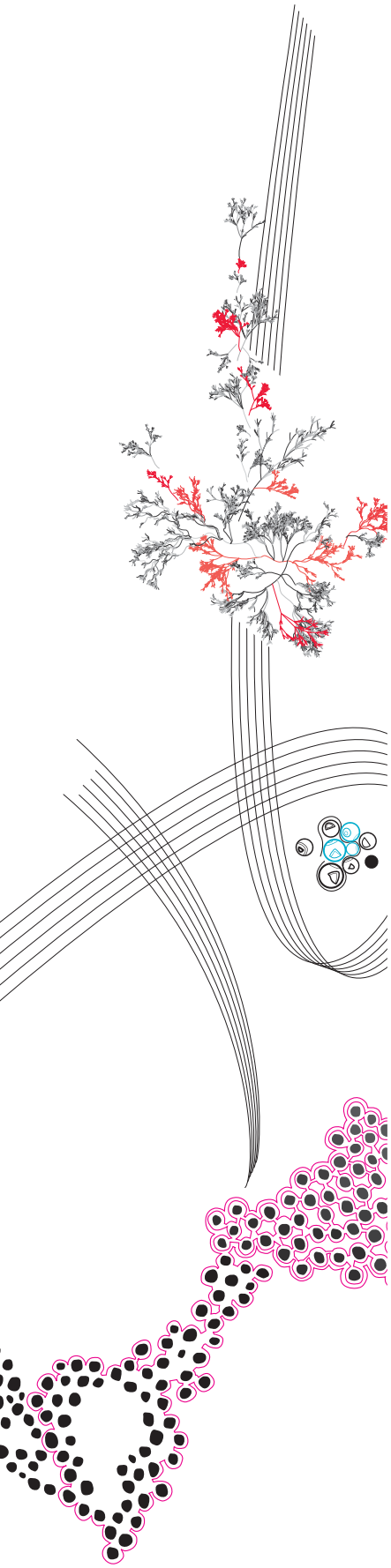

MSc Computer Science
Final Project

Implementing Ensemble Machine Learning Models for Anomaly Detection in Credit Risk Data Quality

Alexandru-Andrei Mădăras

Supervisor: Dr. Luís Ferreira Pires
Dr. Tiago Prince Sales
Dr. Estefanía Talavera Martínez
Can Fensun Oğuz - ING Nederland BV

March, 2025

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Abstract

This thesis explores and compares various anomaly detection methods to enhance data quality issues detection in credit risk analysis for financial institutions. Although various taxonomies exist for anomaly detection methods, this thesis consolidates them into a unified classification based on an extensive literature review. Using this classification, the thesis selects and compares relevant anomaly detection methods from the literature through experiments conducted on a data set from a top European financial institution, ING Bank. These experiments employ parallel (bagging) ensemble learning techniques and leverage Explainable AI to explain the models' decisions to identify specific rows as potential data quality issues.

Keywords: Data quality, credit risk analysis, data quality, anomaly detection, artificial intelligence, ensemble learning, Explainable AI

Chapter 1

Introduction

As businesses accumulate vast amounts of data, the challenge is not merely in collection but in ensuring data quality. As Prashanth Southekal states, “ Companies have tons and tons of data, but [success] isn’t about data collection, it’s about data management and insight." — Prashanth Southekal, author of business analytics, professor, and head of the Data for Business Performance Institute.”

Many businesses and companies shifted their focus to *data quality*, which is a process that lasted for three generations, starting from disconnected data quality checks for business functions before the 1990s to monolithic data silos and ending with the modern period with scalable architecture accessible to people of diverse backgrounds [1].

1.1 Motivation

In the banking industry, this aspect of data quality is influential, especially for making business decisions, in credit risk management, where the risk of poor data quality affects decision-making processes [23]. Moreover, inadequate data quality management can lead to compliance risks under GDPR, impacting an institution’s reputation.

Owning fast, scalable, and reliable data quality tools that can perform quality checks quickly and prevent the propagation of poor-quality data around an organization are solutions that can tackle this problem. Data quality is a crucial factor that can affect the reputation and credibility that society gives to banking institutions. Several companies on the market have started adopting data-driven approaches by implementing data quality tools (models) organized in so-called "data quality management" teams, which implement data quality checks to ensure a more accurate process.

Moreover, given a financial institution’s important role in society, thorough and strict regulations often characterize the way of working inside a bank. These constraints usually imply that innovation attempts to keep up-to-date with current services predominantly suffer, especially in the Tech sector, where changes happen as often as monthly. It also applies to data quality issue detection, where at several banks, the data analysts are currently spending an enormous amount of time using manual, time-consuming, and error-prone data quality checks that can be automated and improved using Artificial Intelligence.

1.2 Problem statement

Data quality is the subdomain of data analysis, which indicates how well a data set complies with the fundamental metrics (accuracy, completeness, timeliness, uniqueness, validity, and consistency) and is crucial for operations inside the banking sector, especially in *credit risk analysis* [13]. However, some institutions still have not transitioned entirely to more automated methods

(using AI), often deciding to stick with manual data management processes. The strict regulations that institutions must follow create a more robust environment for change, causing a slow data management process. *Outlier detection* is the process of detecting data points that can potentially be erroneous, a crucial component for detecting data quality issues. Thus, in specific cases of financial institutions (e.g. within ING Nederland B.V.), we identify several challenges, expressed by the following research questions:

- **RQ1:** Many banking institutions still rely on data analysts for data quality issue detection, resulting in an error-prone and time-consuming process. What are the benefits and challenges of rule-based/manual data quality detection compared to AI-driven approaches in banking institutions?
- **RQ2:** According to [6], many financial organizations use traditional data management approaches involving periodic audits, leading to delays in addressing data quality issues. How can AI-driven approaches accelerate data quality management processes in financial organizations compared to periodic audits?
- **RQ3:** Manual data quality tools often rely on attribute checks without explaining the detected issues. How can Explainable AI improve the interpretability and effectiveness of machine learning models for data quality issue detection?
- **RQ4:** The implementation of ensemble learning algorithms for anomaly detection is limited. What benefits could the adoption of ensemble learning methods for anomaly detection provide in financial institutions?
- **RQ5:** With the various classifications and taxonomies of anomaly detection discussed in Chapter 3, the current classifications are scattered around several criteria, often contradicting each other. How can a standardized taxonomy improve the consistency and cross-domain applicability of anomaly detection methods?

1.3 Objectives

With these problems formulated, this thesis aims to find the appropriate anomaly detection solution based on AI to detect data quality issues within a banking institution's data sets. Based on the study conducted on anomaly detection techniques, this thesis tackles the following research goals:

1. Compare the benefits of using AI-driven approaches replacing rule-based/manual data quality detection in banking institutions.
2. Develop and evaluate a real-time AI-based anomaly detection system to proactively identify data quality issues, reducing reliance on periodic audits and minimizing delays in financial data management.
3. Apply Explainable AI (XAI) techniques to enhance the interpretability of machine learning models for data quality issue detection, ensuring transparency and trust in AI-driven solutions.
4. Implement and evaluate an ensemble machine learning approach to compare the performance of multiple anomaly detection techniques, assessing its impact in the credit risk area.
5. Propose a structured and systematic classification framework for anomaly detection techniques, addressing inconsistencies in existing taxonomies to improve their applicability in financial contexts.

This research explores the benefits of implementing AI models to assist data analysts with manual tasks to detect data quality issues. It offers a promising outlook for the future of data quality detection in banking, with potential improvements in efficiency and accuracy.

1.4 Approach

To address these research questions, the thesis follows a structured approach. It introduces key background terminology related to credit risk metrics and data quality dimensions, followed by a comprehensive review of the literature on machine learning, ensemble learning, anomaly detection techniques, and explainable AI. Before the experiments are conducted, the data preparation and analysis processes and the models' architecture are detailed. The empirical study applies various anomaly detection methods to a banking dataset, comparing different models, assessing their explainability and potential for real-time data quality monitoring. The results are evaluated through internal datasets and expert domain knowledge. Finally, this research proposes a standardized taxonomy for anomaly detection and explores the feasibility of ensemble learning to improve data quality detection performance in banks.

1.5 Methodology

This research follows the *Design Science Research (DSR)* methodology [28], which is particularly suited for studies aimed at developing and evaluating innovative artifacts. The choice of DSR is motivated by the need to create a novel ensemble machine learning model that enhances data quality issue detection within the banking sector. Unlike purely empirical approaches that focus on observational analysis, or theoretical approaches that lack implementation, DSR allows for both model development and validation in real-world settings.

As illustrated in Figure 1.1, the DSR framework consists of the following phases:

1. **Problem Identification:** The research problem is defined through the formulation of research questions, highlighting the limitations of existing data quality detection methods in banking (Sections 1.2 and 1.4).

2. **Objectives Definition:** The research objectives are established to address the identified challenges, focusing on improving anomaly detection models and explainability (Section 1.3).

3. **Investigation of the Current State-of-the-Art:** A comprehensive literature review is conducted to examine existing methods in machine learning, anomaly detection, and Explainable AI, serving as a foundation for developing a novel approach (Chapter 3).

4. **Design and Development of the Solution:** The thesis proposes an artifact in the form of an ensemble machine learning model for data quality issue detection. The model's architecture, preprocessing steps, and selected algorithms are detailed in Chapters 4 and 5.

5. **Experimental Evaluation:** The proposed model is empirically tested on real-world banking datasets, comparing its performance against traditional anomaly detection techniques (Chapter 5).

6. **Evaluation and Validation:** The results are analyzed based on model performance metrics (Recall, Precision, and F1-score), explainability (Shapley value scores), and feasibility for real-time data quality monitoring (runtime). Internal banking datasets and domain expertise are leveraged to ensure the relevance of findings, as discussed in Chapter 6.

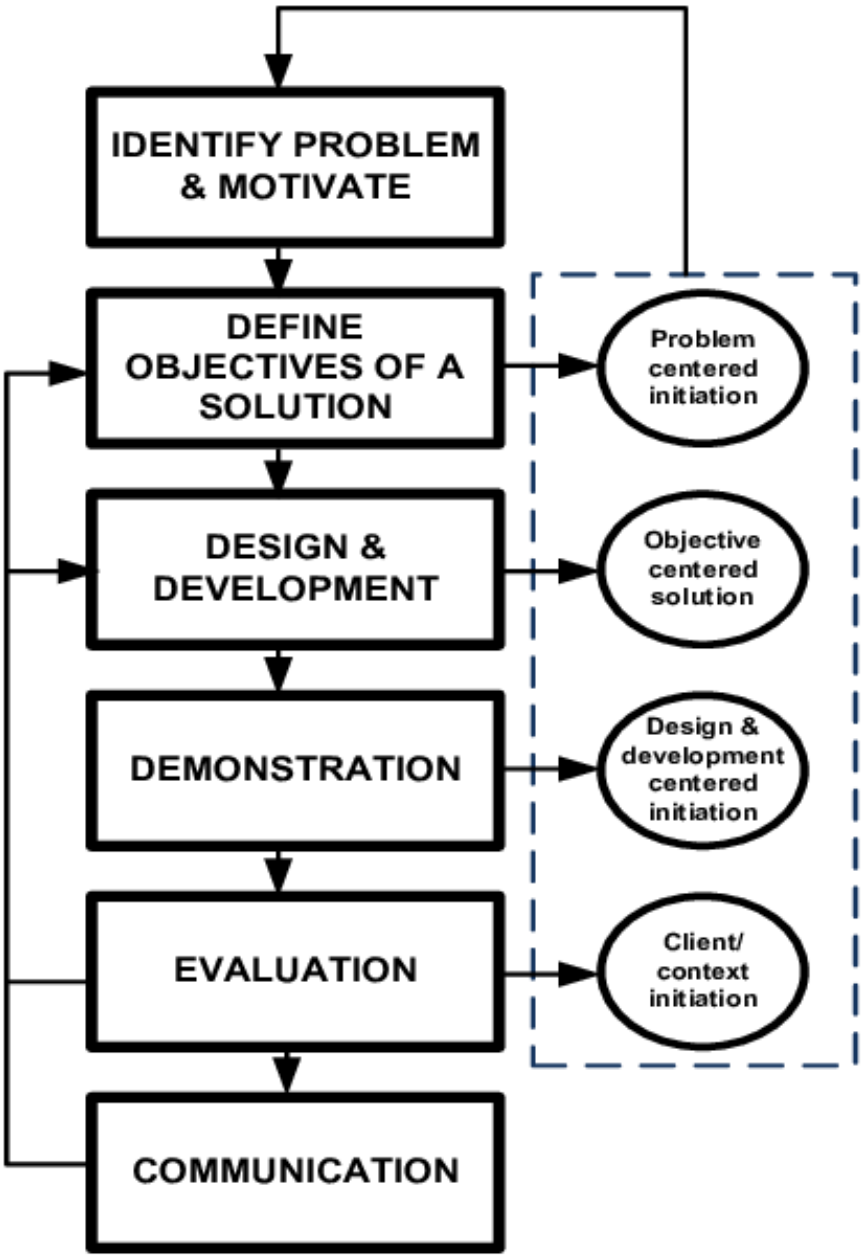

FIGURE 1.1: Design Science Research Process

7. **Communication of the results:** The study's contributions are summarized, along with recommendations for future research directions in AI-driven data quality detection (Chapter 7).

While the primary research methodology is Design Science, an *experimental research* component is integrated into the evaluation phase to quantitatively assess the effectiveness of different anomaly detection techniques. This hybrid approach ensures that the proposed model is not only theoretically sound but also empirically validated in a practical banking environment.

1.6 Thesis structure

This thesis continues with Chapter 2, which introduces and explains the ten metrics used for credit risk analysis and the six data quality dimensions. Chapter 3, where a comprehensive literature study is conducted on *machine learning*, *data quality*, and *Explainable AI*, ends with a *taxonomy development* of anomaly detection methods. Chapter 4 covers the approach conducted through the experiments, then explains in detail the preparation steps performed before the experiments. Chapter 5 explains the architecture of the built models and shows the experiments carried out on ING's credit risk data sets. The results of the experiments are discussed in Chapter 6, ending with the conclusion and future work in Chapter 7.

Chapter 2

Background

This chapter discusses the core concepts *credit risk*, introducing the ten important credit metrics used for risk analysis. In the second section of this chapter, the data quality concept is introduced and discussed from the banking perspective.

2.1 Credit risk management

Credit risk analysis represents a crucial process performed in financial institutions to ensure the efficiency of loan portfolio management by evaluating the creditworthiness of potential borrowers [38]. Credit risk includes the possibility of a client not meeting his obligations and requires prudent oversight to maximize the returns from a loan [9].

Bad credit risk management mainly caused the financial crisis; hence, instruments to control credit risk are crucial. *The Basel Accords* were established in 1988, and they now have three versions, with the most recent one published after the economic crisis of 2007-2008. Their main goal is to ensure that the banking institutions maintain enough cash reserves to meet financial obligations, identify poor corporate governance, liquidity management and the lack of restrictions regarding the over-levered capital structures, which led to a crisis. Moreover, Basel III added new requirements and measures to increase the banks' reserves when credit increases while decreasing the requirements when the lending is lower.

Probability of Default (PD)

PD is a credit risk metric that expresses the likelihood that the borrower will default in a specific period (usually a year). This metric is calculated per loan and estimated using credit ratings, historical default rates, or statistical models. PD is essential for quality credit risk management by evaluating the risk of each portfolio to which the capital is correctly allocated [39].

The calculation of the PD of an obligor is dependent on two sources: internal data and internal model ratings, as well as external data from entities such as Moody's and S&P (see Table 2.1). The cells highlighted with green represent an acceptable grade of credit risk for an institution, while the red cells a high probability of default from the borrower (high credit risk).

Loss Given Default (LGD)

LGD is a crucial metric for *credit risk analysis* that represents the percentage of the Exposure at Default (EAD) lost in the event of a default. Calculating LGD involves addressing several key questions [37]:

- How much is recovered and from where (e.g., collateral liquidation)?
- What is the duration of the recovery process, and what financial costs (e.g., forgone interest income) are associated with it?

TABLE 2.1: Moody's, S&P's, and Fitch's Rating Grades [5]

MOODY'S	S&P	FITCH	DESCRIPTIONS	GRADE
Aaa	AAA	AAA	Highest credit quality, minimal credit risk	
Aa	AA+	AA+	Very high credit quality, very low credit risk	Investment Grade
A	A, A-	A,A-	High credit quality, low credit risk	
Baa	BBB	BBB	Good credit quality, moderate credit risk	
Ba	BB	BB	Issuer faces adverse conditions and uncertainty, substantial credit risk	
B	B	B	High default risk, issuer able to meet financial commitments	
Caa	CCC	CCC	Vulnerable, default likely	High Yield
Ca	CC	CC	Issuer is highly vulnerable or near default	
C	C	C	Lower ratings, issuer in default	
	D	RD D		

- How much was spent during the recovery process (e.g., workout expenses)?

LGD is the opposite of the Recovery Rate (RR), which represents the proportion of the total exposure recovered after a default event. Both metrics are expressed as a percentage, where LGD is calculated as [9]:

$$LGD = 1 - RR \qquad RR = \frac{\text{Recovered Amount (RA)}}{\text{Exposure at Default (EAD)}} \tag{2.1}$$

where RR stands for the recovery rate.

Exposure at Default (EAD)

Exposure at Default measures the potential bank's loss if a borrower defaults during their credit. The loss relies on the extent of the bank's exposure to the borrower at the instant of Default, an event that could happen at an unpredictable point.

According to Basel, it is calculated with the following expression [9]:

$$EAD = CurrentExposure + LEF \cdot UnutilisedPortionOfTheLimit \tag{2.2}$$

where LEF is the Loan Equivalency Factor, which represents the amount of the credit line that is not utilized and is expected to be drawn before the borrower defaults. In crises (at Default), the LEF and EAD have an increased utilization, illustrated by Figure 2.1

Expected loss (EL)

Expected loss (EL) represents the statistically measured, long-run average loss level due to credit defaults. For example, consider a bank that expects around 1% of its loans to Default annually,

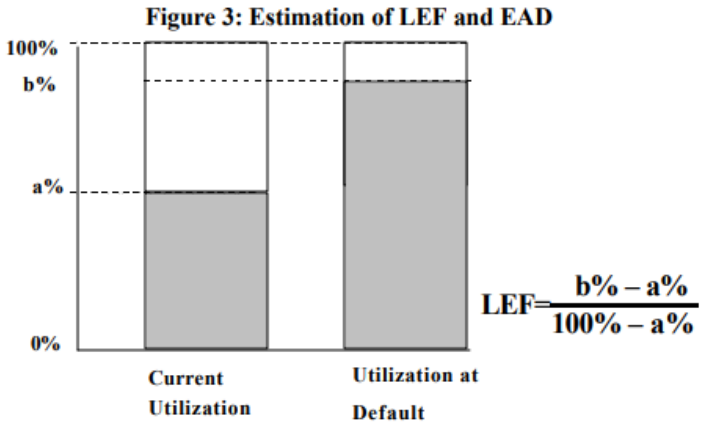

FIGURE 2.1: EAD and LEF utilization normal vs Default

with an average recovery rate of 50%. For a credit portfolio of $1 billion, the one-year EL is calculated as [9]:

$$EL = PD \cdot LGD \cdot EAD \tag{2.3}$$

where

- PD - Probability of Default already discussed before
- EAD - Exposure at Default already discussed before
- LGD - Loss Given Default discussed in this section

Unexpected loss (UL)

Besides anticipating a loss, banks must always store extra funds for safety against unpredictable losses that can extend over the average losses in the past. *Unexpected loss (UL)* aims to measure this aspect and calculates it with the following formulas:

$$UL = \sqrt{PD \times \sigma_{LR}^2 + LR^2 \times \sigma_{PD}^2} \tag{2.4}$$

[9] where:

- PD - Probability of Default already discussed before
- LGD - Loss given Default already discussed before
- $\sigma_L^2 GD$ - Variance of Loss Given Default $= LGD(1 - LGD)/4$ [11].
- $\sigma_P^2 D$ - Variance of default probability $= PD(1 - PD)$ [11].

From the statistical point of view, we can define Unexpected Loss (UL) as the standard deviation of Expected Loss (EL), depicted by the graph in Figure 2.2.

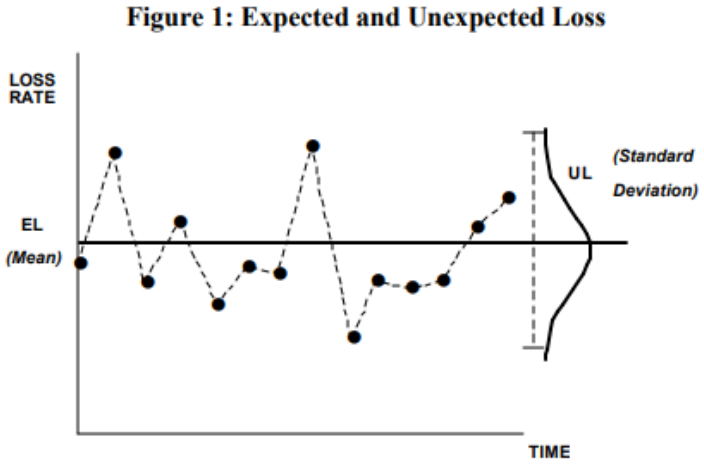

FIGURE 2.2: Expected Loss and Unexpected Loss[37].

Effective Maturity and Maturity adjustment(M)

In Credit Risk Analysis, *effective maturity* refers to the length of time until the principal amount of a loan or financial instrument is due to be paid back. It is expressed as a decimal number, usually represented in years (e.g. a loan with a maturity of 5 years and 6 months can be expressed as 5.5). [3] *Maturity adjustment (M)* metric is introduced to reflect the possible deterioration of credit quality for loans with higher maturities (longer-term loans). It is expressed as a factor (decimal notation) that adjusts for the increased risk of the longer loan maturities (higher than 2.5 years) [37].

Capital Buffer (CB)

Following the guidelines of the Basel II documentation [2], the capital buffer of an institution, also known as regulatory capital, which measures the credit risk, can be determined using four risk parameters: PD (Probability of Default), Loss Given Default (LGD), Exposure at Default (ED), and the Maturity Adjustment (M) factor for loans with effective maturity higher than 2.5. Equation 2.5 defines how the Capital Buffer is calculated:

$$CB = PD * LGD * EAD * M \tag{2.5}$$

Risk weighted assets (RWA)

RWA is the base concept behind the Basel Agreements and is usually the key metric in all credit risk datasets. It sets weights for each loan and impacts a bank's capital adequacy. It aims to guide banks to more prudent lending decisions. Due to limitations, such as the inability of RWA to reflect actual risk exposure, Basel II was introduced in 2004. It aimed to strengthen the banking system by making the capital requirements more robust, which meant changing the definition of RWA. From Basel I continuously, the minimum ratio of regulatory capital to total RWA is set to 8%, but the 'core capital' element (more restrictive definition of eligible capital known as Tier 1 capital) has been set to be at least 4% according to Basel II [37].

Basel II consists of standards used to improve risk management practices, structured into three pillars (elements):

- Pillar 1, addressing minimum requirements for credit and operational risks
- Pillar 2, guiding the supervisory oversight process

- Pillar 3 forces banks to disclose key information on their risk profile and capitalization to keep a market discipline

Using the Vasicek model, the following RWA formulas were derived [37]:

1. **Correlation** (ρ) for corporate, sovereign, and bank exposures - used as a function of PD, ignoring potential important portfolio characteristics such as industry and geographic diversification:

$$\rho = 0.12\frac{\left(1-e^{-50\cdot PD}\right)}{(1-e^{-}50)} + 0.24\left(1-\frac{1-e^{-50\cdot PD}}{1-e^{-50}}\right)^2 \tag{2.6}$$

2. **Maturity Adjustment** (b) - introduced to reflect potential credit quality (PD) deterioration for loans with longer maturity:

$$b = (0.11852 - 0.05478 \cdot \ln(PD))^2 \tag{2.7}$$

3. **Capital Requirement** (K) :

$$K = (\text{LGD} \cdot \phi(\frac{\phi^{-}1(PD)}{\sqrt{1-\rho}} + \sqrt{\frac{\rho}{1-\rho}} \cdot \phi^{-}1(0.999)) - PD \cdot LGD) \cdot \frac{1+(M-2.5)\cdot b}{1-1.5\cdot b} \tag{2.8}$$

4. **Risk-Weighted Assets** (RWA):

$$RWA = K \cdot 12.5 \cdot EAD \tag{2.9}$$

5. **Capital Charge**:

$$\text{Capital charge} = 8\% \cdot RWA \tag{2.10}$$

where

- $\Phi^{-1}(z)$ is the inverse cumulative distribution function (c.d.f.) for a standard normal variable z.
- $\ln(PD)$ is the natural logarithm PD.
- $\Phi(y|\rho, v)$ is the c.d.f for a standard normal random variable y.
- v is the value of v such that $\Phi(v) = y$.
- M is the effective (remaining) maturity.

First, the capital formula incorporates loss correlation but models it solely based on the Probability of Default (PD). Moreover, it overlooks crucial portfolio characteristics such as industry and geographic diversification, relegating them to Pillar II (supervisory review).

The formula 2.7 expresses the *maturity adjustment*, a term that accounts for potential credit quality deterioration in loans with longer maturities. Here, the average portfolio's effective maturity is anchored at 2.5 years.

Risk weights are adjusted to ensure that a bank maintains sufficient capital to cover unexpected credit losses with a 99.9%

From the literature, [23] shows the relation between credit risk and data quality. The paper focuses on the DQ dimensions assessed using a scorecard index, where the DQ challenges are tackled. Moreover, the study underscores the importance of maintaining high data quality for accurate credit risk assessment and adequate decision-making in financial institutions. It further suggests implementing a Total Data Quality Management (TDQM) program. The paper then highlights the context-dependent nature of DQ and presents an empirical study to assess the DQ level of credit risk databases. The extended to additional data quality dimensions, namely actionability, security, relevancy, objectivity, value-added, and representational consistency, where the conclusion is that the enhancement can be focused on the relationship between credit risk modelling parameters used to calculate the buffer capital (BC) with Formula 2.5.

2.2 Data Quality Management

Data Quality Management is one of the most important aspects of any current data-driven process, being classified by the level it is applied: on macro and micro-level. On macro-level the focus is on examination and improvement of processes that affect data quality, while at the micro-level the focus is on the database aiming to repair errors and artefacts [4].

Data quality impacts crucial processes of businesses and institutions. A notorious example is the US government, which, according to IBM, lost $ 3.1 trillion yearly due to bad-quality data. Another example is MIT, where employees manage data quality tasks half their time. There are several definitions for data quality in the literature. According to Wikipedia, *data quality* in the banking industry covers a set of techniques, algorithms, and methodologies to evaluate and improve the quality of the financial data sets [25]. According to Moges et al., [23], data quality implies using standards that ensure a financial company makes data-driven decisions to fulfil its business goals.

There are six dimensions of data quality against which a dataset is assessed in the context of credit risk data:

1. **Accuracy** - Ensures that a dataset has accurate values, measured by calculating the correctness of numerical and categorical values. Numeric values are gauged by the percentage of outliers (anomaly points) indicating inaccuracies.

2. **Completeness** - Ensures that a dataset has no missing records in mandatory fields, measured using the ratio of missing values.

3. **Validity** - Measures the percentage of data conforming to the required format for specific business rules. Data is valid if values follow specified types, ranges, patterns, etc, defined by the financial institution.

4. **Consistency** - Evaluates records from different datasets to check their relationship based on a standard source-of-truth table. Consistency is measured by calculating the percentage of corresponding records from other tables with differing values. This dimension is crucial for maintaining trust in an organization's dataset.

5. **Uniqueness** - Ensures that a dataset contains no duplicate primary keys, evaluated by computing the percentage of duplicates.

6. **Timeliness** - Requires the dataset to be regularly updated. It is measured using the frequency distribution of dates and specific bank rules and is often assessed on time series data.

Any value which does not comply with one of the *data quality* (DQ) dimensions' definition constitutes a *data quality (DQ) issue*. Our study also refers to these data points as *anomalies* or *outliers*. From the perspective of occurrence, data quality issues can be split into two categories [36]:

1. Random - if the DQ issue (anomaly) occurs due to a random error in the system that does not relate to other data quality issues.

2. Systemic - if the error repeats throughout the data set systematically. In this case, an expert can solve the issue by removing the related cases using data manipulation or other specific techniques.

Additionally, Tony Ho et al. [6] addresses critical challenges financial institutions face in maintaining robust data control capabilities, particularly emphasizing the importance of data quality, data lineage, and transaction testing. These challenges are compounded by the need

for regulatory compliance and the complexities associated with various data requirements. The paper highlights the need for banks to adopt standard and reusable solutions to meet these stringent requirements effectively.

In this context, this thesis proposes the implementation of ensemble machine learning models, as a solution to enhance data quality management within credit risk assessment. By leveraging the strengths of these models, the proposed approach aims to improve anomaly detection accuracy, thereby ensuring more reliable and accurate data for financial decision-making.

Chapter 3

Literature review

This chapter represents the third step of the Design Science methodology (Section 1.5 - Investigation of the Current State-of-the-Art), which introduces and discusses the concept of AI-driven Anomaly detection. The discussion starts with introducing *machine learning algorithms*, *ensemble learning technique*, and continues with a classification of these methods, which aims to standardise the ones from literature. The chapter ends with a classification and a detailed description of the selected *anomaly detection* methods.

3.1 Machine Learning

Machine learning is defined as a learning technique that aims to improve the performance of a system using algorithms that build models from data, where the human is not involved. By applying specific learning algorithms, the model is trained on a particular dataset, such that it can predict a value using labels (e.g. predict the price of an estate) or without labels (e.g. detecting anomalies in data) or delayed feedback with a specific goal (e.g. image processing). As part of Artificial Intelligence, Machine learning is a technique based on algorithms trained on datasets to create models to perform tasks that are difficult to complete for humans. From this perspective, the *anomaly detection algorithms* can be classified by taking into account the trained dataset as supervised and unsupervised, also commonly referred to as "supervised anomaly detection" and "unsupervised anomaly detection".

Supervised learning is a machine learning approach that uses labelled datasets to classify data or to predict outcomes accurately. Using labelled inputs and outputs, the model can measure its accuracy and improve over time [7]. Two central problems can be solved using supervised learning: classification and regression.

Unsupervised learning, on the other hand, does not use labelled data provided by humans and is called unsupervised for this reason. It solves problems related to clustering, association, and dimension reduction.

Another type of learning called *semi-supervised learning* combines supervised and unsupervised methods. It is primarily used when unlabelled data is already present while getting the labelled data is slow.

Lastly, there is also the *reinforcement learning* technique, which aims to maximise some notion of cumulative reward using software agents [21].

3.2 Ensemble Learning

Ensemble learning is a Machine Learning technique that combines multiple learners (models, neural networks, algorithms) to provide better predictions than one model. The method aims to solve the bias-variance trade-off by combining several models to maintain their particularities

and benefits while reducing the error rate. On top of this, ensemble learning helps resolve high-dimensional data issues by offering an alternative to it.

The ensemble method enhances predictive performance by averaging the variance and errors associated with individual models, thus creating a more robust and accurate predictor. For instance, Isolation Forest, which excels at managing large datasets with numerous features, can highlight feature importance. At the same time, an SVM can find complex boundaries between classes, even in high-dimensional spaces. By integrating their predictions, typically through techniques like majority voting or stacking, the ensemble can mitigate noise and improve overall prediction accuracy without dimensionality reduction, demonstrating resilience and robustness in different tasks.

According to the literature [14], there are two groups of ensemble learning methods:

- Parallel - where the base learners are trained in parallel and independently
- Sequential - where each model is trained once at a time and where the next one minimises the errors made by the model trained in the previous step.

Three popular ensemble learning techniques that implement these groups and are used in this thesis are:

1. Bagging - a homogenous parallel method replicating one training data set to train multiple models using the same algorithm. A typical example is random forest, an extension of bagging because it splits the training sets into random subsets of features to create a decision node [43].
2. Stacking - a heterogenous parallel method that trains several models using different training algorithms for each learner. The model predictions are compiled and used to train the resulting model, also known as the meta-model [10].
3. Boosting - use sequential order to train each model. The first model performs poorly, so the next one in the chain trains the output produced by the first learner with a focus on the misclassified data. There are two types of boosting depending on their prioritisation: *adaptive boosting (AdaBoost)*, which adds weights to the misclassified samples of the previous learner, and *gradient boosting*, which uses residual errors from the learner to set the target prediction of the following learner [17].

3.3 Anomaly Detection

Before discussing anomaly (outlier) detection, a clear definition of the concept of anomaly (outlier) should be introduced, being the foundation of the anomaly detection concept.

3.3.1 Outliers

Outliers are data patterns that do not correspond to the notion commonly established as "normal behaviour" [34]. They can appear as points or context, depending on the type of representation. Thus, based on the literature [34], we classify anomaly detection using the criteria of outlier types as point, contextual, and collective.

Point Outliers

Point outliers focus on data points that lie outside the boundaries of a region with regular points. For example, fraudulent credit card detection, where an enormous amount is spent compared to the history of the suspect's transactions, is considered an anomaly point.

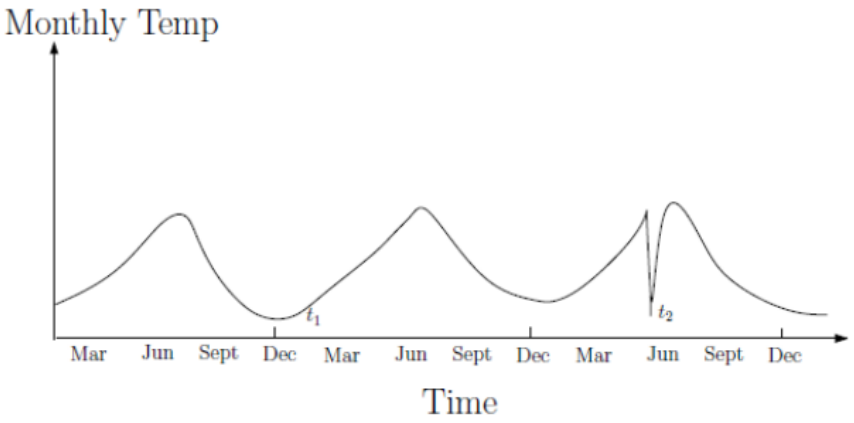

FIGURE 3.1: Contextual Outliers on Time-series data [34]

Contextual Outliers

Contextual outliers are anomalous data instances (points) in a specific context. They are of two types: contextual parameters, used to determine the context (neighbourhood, e.g., latitude and longitude), and behavioural attributes, used to define non-contextual properties of an instance (e.g., humidity of the entire world). Both types are explored in time series and spatial data and require a clear context definition. Given the behavioural attributes within a particular context, an instance is classified as an anomaly or normal. For example, a point can be an outlier in one context but a regular point in another. Thus, this method requires identifying contextual and behavioural attributes for contextual anomaly detection. Figure 3.1 illustrates this behaviour on time-series data, using *time of purchase* as a contextual attribute. Applying this detection method depends on the relevance of the contextual outliers in the target domain. Therefore, the context must be well-defined to achieve high performance in detecting anomalies with this method.

Collective Outliers

Collective outliers represent a collection of data instances (points) that represent anomalous properties compared to the rest of the data set cases. This type of outlier extends the definition terminology to an entire group of data points or a pattern representing an anomaly, known as *collective anomaly*. This type of outliers is explored in sequence, graph, and spatial data. A specific example is an electrocardiogram, where the Atrial Premature Contraction is considered the anomaly, represented by a sequence of points that shape a P wave, illustrated by Figure 3.2. Given that context, outliers depend on the ability to define a context for detecting anomalies; a collective outlier can also be considered contextual if the context is determined.

3.3.2 Classification of Anomaly Detection Methods

Considering the multiple classifications discussed in the literature [31] [34] [27], this thesis proposes a general classification that aims to serve as a consensus of the existing ones. Thus, we classify *anomaly detection* from three perspectives: *methodology*, *technique*, and *outlier type*, illustrated by Table 3.1.

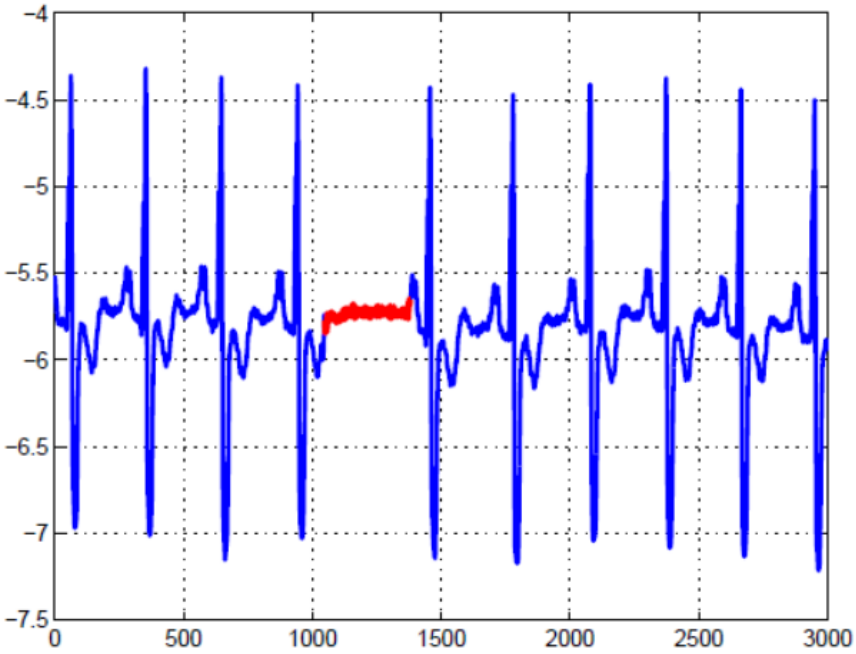

FIGURE 3.2: Collective Outliers in a human ECG corresponding to an Atrial Premature Contraction [34]

TABLE 3.1: Classification of Outlier Detection Techniques

Perspective	**Category**	**Examples**
Methodology	Statistics-based	Z-score, Inter-quartile range (IQR), Principal Component Analysis (PCA)
	Machine Learning-based	Decision Trees, SVM, KNN, Random Forests, Isolation Forest, Bayesian Networks, Hidden Markov Model
	Deep Learning-based	CNN, RNN, LSTM, Autoencoders, GANs
	Data Mining-based	Clustering, Classification-based outlier detection
Outlier Type	Point	Sudden spike in network traffic, high bank transaction
	Contextual dependent	context are well-identified before applying the method. e.g. time of purchase in credit card fraud detection [34]
	Collective	Sequence of fraudulent transactions, cluster of failed logins
Technique	Statistical Models	Gaussian Distribution Models
	Density-based	LOF, RDF, DWOF
	Distance-based	K-nearest Neighbors
	Clustering-based	CBLOF, CBOD
	Isolation-based	Isolation Forest

Dimension	Category	Examples
	Ensemble-based	LODA, LSCP
	Subspace-based	SOD, LSOF, COP

3.3.3 Method selection

Given the high number of methods introduced in the classification from Table 3.1, we further researched to identify the strengths and weaknesses of the most significant methods, illustrated by Table 3.2.

TABLE 3.2: Comparison of the algorithms discussed in this report

Algorithm	Type	Advantages	Disadvantages
Autoencoders	Deep Learning	• Dimensionality reduction • Flexibility • Detection of novel anomalies • Scalability (good for high dimensional data) • Unsupervised - no labelled data required	• Requires a large amount of training data • Hyperparameter sensitive • Overfitting risk - can be computationally intensive
GANs	Deep Learning	• Flexibility • Good for high-dimensional data • High accuracy- able to self-generate fake data • Excellent for detecting subtle anomalies • Unsupervised - no labelled data required	• Require complex architecture • Hyperparameter sensitive - difficult to train • Computationally intensive • Interpretability - hard to interpret its results

Continued on next page

Table 3.2 – *Continued from previous page*

Algorithm	Type	Advantages	Disadvantages
Isolation Forest	Machine Learning	• Good for high-dimensional data • Excellent runtime • High detection accuracy • Robustness - focus on isolating anomalies	• Consistency - can be inconsistent in determining outliers in certain scenarios • Performance on lower data sets - may perform poorly on medium-sized datasets and linearly separable classes
k-Nearest Neighbor	Machine Learning	• Simple to implement • Good on small datasets	• Sensitive to noise • Computationally expensive for large datasets
LOF	Machine Learning	• Local Density-Based Detection - Effective for unsupervised outlier detection • Robustness - focus on local density • Flexibility - does not require labels	• Computational Complexity • Parameter Sensitivity • Interpretability - their results could be hard to interpret • Scalability - May perform poorly on high-dimensional data
Bayesian Networks	Machine Learning	• Effective for anomaly detection by identifying deviations from expected patterns • Useful in diagnosing faults and predicting failures	• Requires domain knowledge for model construction • Can be inconsistent in determining performance parameters

Continued on next page

Table 3.2 – *Continued from previous page*

Algorithm	Type	Advantages	Disadvantages
Histogram methods	Statistical non-parametric	• Simple • More flexible and autonomous than parametrical methods • Perform well on low dimensional data	• Requires prior knowledge for learning the distance-based similarity • Performs poorly on high-dimensional data
Simple outlier detection technique	Statistical parametric	• Simple, quick • Perform well on numerical, univariate and normally distributed data	• Perform worse on randomly distributed data • May miss subtle anomalies due to its simplicity

Due to the domain of application for the solution in banking, where the data sets are almost always enormous, we selected the following methods as possible implementations:

- **Isolation Forest** - given its simple structure, it goes without saying that it is the best algorithm in terms of running time. Moreover, it can handle high-dimensional data due to its random feature selection and linear complexity, making it efficient for this case. Last, given its robustness, it isolates anomalies using feature selection and split values, which increases its potential to detect anomalies accurately. Given its run time and accuracy, it can be well combined with Autoencoders.
- **Autoencoders** - due to its dimensionality reduction capability, it can successfully handle high dimensional and large data sets. Given its dimension-reduction ability, it could prove efficient when combined with Isolation Forest in an ensemble of models.
- **Generative Adversarial Networks (GANs)** - given their complex structure, which can handle high-dimensional data, it proves a well-fitting choice for large data sets. Last but not least, having two components that train each other can significantly enhance the accuracy of the ensemble model when detecting subtle anomalies. It is also unsupervised, meaning it doesn't require labelled data.

3.4 Isolation Forest

Isolation forest is an *unsupervised machine learning algorithm* that builds an ensemble of *isolation trees* for a given data set, where the anomalies are considered the instances with short average path lengths on the isolation trees [19]. The method is based on graph theory, incorporating the concept of nodes and binary trees. Thus, the following terms are introduced:

Definition: Isolation Tree (iTree). 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 descendent 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 [19].

Thus, if we take a tree T and divide it into external (leaves) and internal nodes, the non-leaf nodes are based on an attribute q, the split value p of attribute q, and two child nodes: (T_l, T_r),

where p is a random value between the maximum and minimum values of the attribute q. These two literals (p,q) determine the separation of nodes based on the comparison of q value, which determines whether the new sample is classified as T_l or T_r [41].

Definition: 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 [19].

In the training step, if the data is d-dimensional as the set $X = \{x_1, x_2, ..., x_n\}, X' \subset X$ is computed by sampling X with size ψ. The method works recursively by splitting X' until each node contains only one sample or all samples of a node have an identical value. Then, the dataset is sampled n times to train different isolation trees, and by merging them, it constitutes the ensemble isolation forest method [41]. The path length determines the degree of susceptibility to isolation:

- a short path length represents a high probability of isolation
- a long path length represents the low probability of isolation

According to Zhang [41] *anomaly detection* using Isolation Forest is composed of two stages: *training stage*, and the *evaluation stage*.

Training stage

In the first stage, *isolation trees* are built recursively by partitioning a sub-sample X' until all the instances are isolated. The algorithm has two input parameters: sub-sampling size (ψ) and the number of trees t. The algorithm's output is a collection of trees, further evaluated in the second step.

Evaluation stage

In the second stage, the length of the paths is calculated by counting the number of edges e from the root node to the instance node x. When the traversal reaches a predefined height limit $hlim$, the algorithm returns the number of edges e plus an adjustment of c. The adjustment represents the average path length of a random sub-tree that could be constructed using data whose size is beyond the tree height limit. The *anomaly score*, calculated with the equation below, is obtained by computing the single path length $h(x)$ and has values in the interval $[-1, 1]$ after normalisation.

$$s(x, \psi) = 2^{\frac{E[h(x)]}{c(\psi)}} \tag{3.1}$$

Here:

- s(x, ψ)) represents the anomaly score of the data point x, given a set of ψ instances
- (E[h(x)]) denotes the expectation (average) of the function h(x) from a collection of isolation trees.
- (c(ψ)) is the average of h(x) given ψ, used to normalize h(x).

After the normalisation, the anomaly score is interpreted as *anomalous* if the instances return a score s very close to 1 and as *normal* if the cases have a score much more minor than 0.5. The dataset is balanced without any anomaly if all the data points are around 0.5. Figure 3.3 illustrates the visualisation of the *isolation forest* method using a contour, where the potential anomalies are the instances with a score $s > 0.6$.

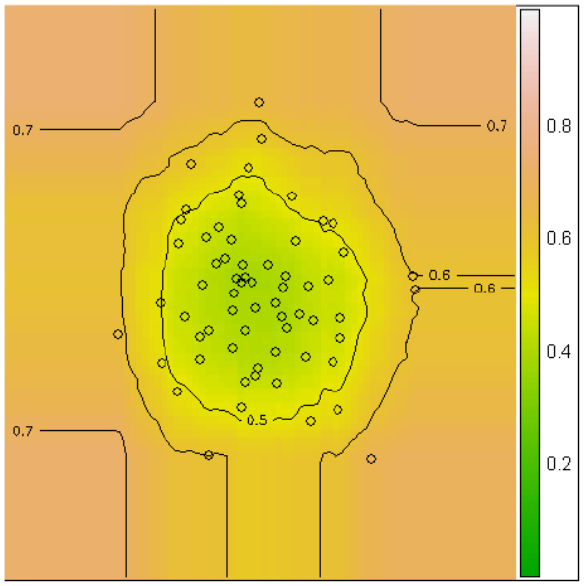

FIGURE 3.3: Anomaly score contour of isolation forest for a normal distribution of 64 points [19]

With a relatively low time complexity of $O(n \cdot t)$, the algorithm can be further applied for detecting *data quality issues*. For *accuracy*, *validity*, and *completeness*, the algorithm detects data points with missing values by identifying unique paths in the isolation trees, leading to shorter path lengths, and thus can be identified as anomalies.

For *consistency*, one can create separate Isolation Forest models for each data source. If the models give significantly different anomaly scores for the same input data, it indicates a potential inconsistency issue.

The algorithm might not be directly applied to *timeliness*, but one can use it together with time-series analysis. For example, one could use Isolation Forest to detect anomalies in the distribution of data timestamps, which could indicate out-of-date data.

Isolation Forest can also help detect duplicate data. If a data point duplicates another, they will have similar paths in the isolation trees, leading to longer path lengths. However, other methods could prove more suitable for detecting duplicate records because the algorithm is typically used to detect outliers, not *uniqueness* related issues.

3.5 Autoencoder

Autoencoder is a *deep learning algorithm*, consisting of feed-forward Neural Networks with multiple layers that transform an input vector into an output vector in an *unsupervised* manner. A popular benefit of using this specific deep-learning method is its ability to reduce the dimensions of a data set. Formally, autoencoders are defined as follows:

Definition. Autoencoders

An *autoencoder* is an algorithm aiming to learn an "informative" representation of the data that can be used for different applications by learning to reconstruct a set of input observations well enough [22].

Structurally, an autoencoder has three components:

1. **Encoder** - maps the input data into a lower-dimensional code (latent space representation) and extracts the essential features from the input.

2. **Latent feature representation** - contains the essential information from the original

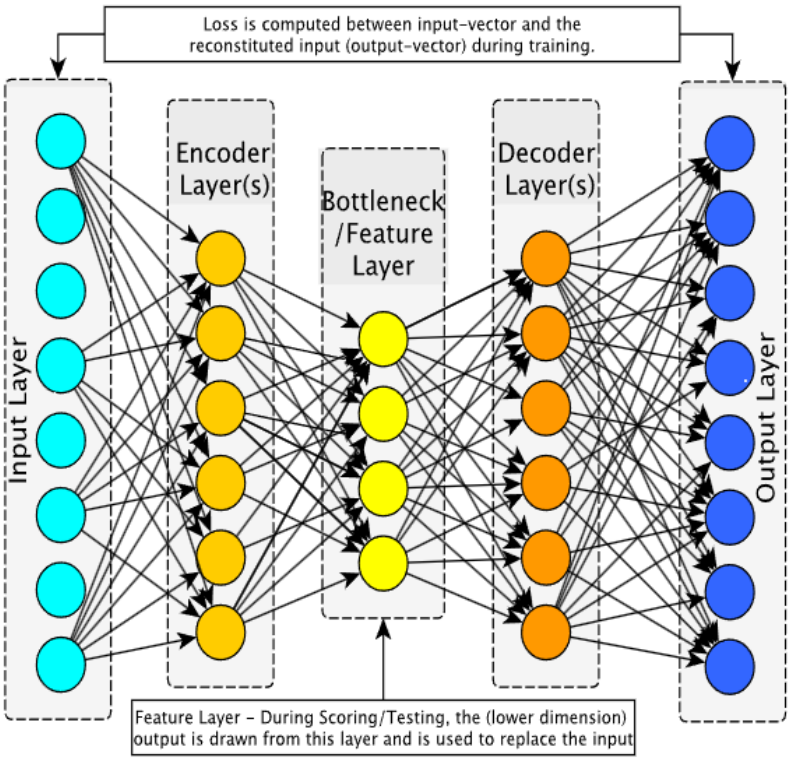

FIGURE 3.4: Autoencoders' architecture - schematic illustration [33].

input encoded into a tensor of real numbers

3. **Decoder** - reconstructs the original input from the latent feature representation.

These components are implemented using Neural Networks and trained using the *back-propagation* function. Figure 3.4 illustrates the main components of a Neural Network adapted to the *autoencoder's* structure. Here, the structure is made of the input layer, output layers, and hidden layers. The latter ones are made of the encoder layer(s), the latent feature representation (on a lower dimensionality) illustrated by the bottleneck layer, and the decoder layer(s). An autoencoder can have one or multiple encoder/decoder layer(s), depending on the use case and the complexity of the data. Back-propagation is the core of a feed-forward neural network because it determines which neural network weights must be adjusted during each iteration of the training phase. The encoder can be written as a function g, which depends on the parameters:

$$h_i = g(x_i) \tag{3.2}$$

, where $h_i \in \mathbb{R}^q$ is the latent feature representation calculated as the encoder output evaluated on the input x_i (the previous neural network layer). The decoder can be written as a composed function f of the latent features:

$$\tilde{X}_i = f(h_i) = f(g(x_i)) \tag{3.3}$$

, where $\tilde{X}_i \in \mathbb{R}^n$ is a real number. The training algorithm of autoencoders requires finding the functions g and f that satisfy the equation:

$$\arg\min_{f,g} \langle |\Delta(x_i, f(g(x_i)))| \rangle \tag{3.4}$$

Where Δ measures the difference between the input and the output, which in Neural Networks translates to the loss function between the input and the expected value (in this case - the output)

$\langle . \rangle$ represents the average of all observations. Another component of Figure 3.4 is *the bottleneck layer*, which illustrates the lower dimensionality than the input. The reason for including this is the possibility of perfect reconstruction of the output, which means learning the identity function. On top of this, the regularisation method is applied to deal with the overfitting problem. To achieve this, two regularisation parameters are added to the final result of the loss function:

$$\arg \cdot \min_{f,g} |\Delta(x_i, f(g(x_i)))| + \lambda \sum_i \theta_i^2 \tag{3.5}$$

, where θ_i are the parameters in the functions f, g (parameters are the weights).

While traditional auto-encoders have a fixed latent space that is deterministically represented, variational auto-encoders represent their latent space as probabilistic, modelled as a multivariate Gaussian distribution, allowing for sampling and creating new data points.

As a deep learning algorithm, autoencoders have activation functions crucial for anomaly detection. The most popular activation functions are *ReLU* and *Sigmoid function*. *ReLU* assumes all values in the range [0, inf], given the formula:

$$ReLU(x) = max(0, x). \tag{3.6}$$

Choosing ReLU is only beneficial when the input values x_i have positive values. *Sigmoid function* σ assume all values are in the range [0,1], and has the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$

According to their structure, there are six types of autoencoders [30]:

1. **Vanilla autoencoders** - contain just an encoder and a decoder, where training minimises the reconstruction error using back-propagation.

2. **Sparse autoencoders** - use a regularisation term to enforce sparsity in the hidden layer. Hence, the number of active neurons is reduced, and only one hidden layer is used. This architecture is achieved by introducing a sparsity constraint that pushes the activations of the hidden layers close to zero.

3. **Denoising autoecnoders** - add noise to the input data and learn to recover the original data, improving the model's generalisation ability.

4. **Contractive autoencoders** - use a penalty term to make the hidden layers more robust to slight variations in the input data, thus encouraging feature extraction.

5. **Stacked autoencoders** - formed by stacking multiple layers of autoencoders on top of each other, hence having multiple hidden layers.

6. **Variational autoencoders (VAEs)** - use uncertainty modelling to learn meaningful latent representations with approximate posterior distributions for better data matching.

7. **Recurrent autoencoders** - incorporates recurrent layers for sequential data (time series).

Anomaly Detection using auto-encoders

There are multiple types of autoencoders for anomaly detection, the most popular being variational autoencoders, denoising, and deep autoencoders. Kumar et al. [16] used a Machine Learning model to detect data quality issues through two components: *a predictor*, and *a splitter*. An autoencoder is used in the first stage, where after it passes the data entered by the user to the already trained model, it applies dimensionality reduction on the data set and detects the anomalies.

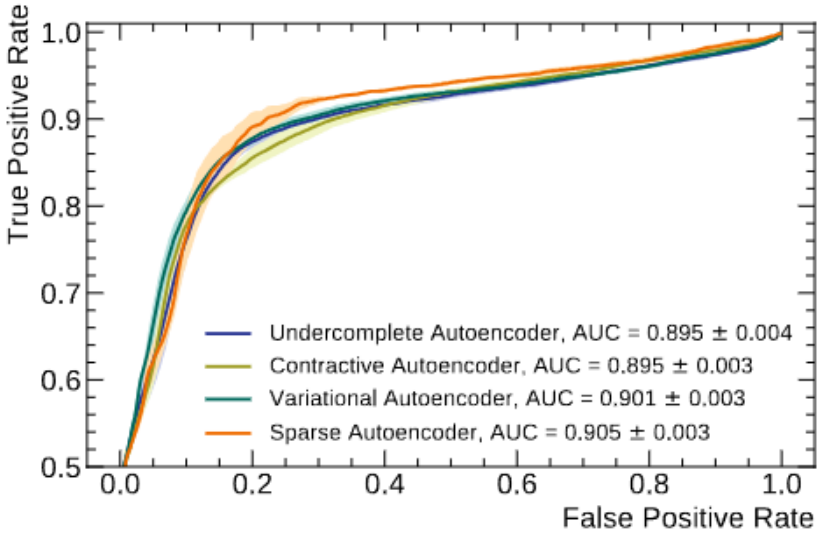

FIGURE 3.5: Area Under the Curve (AUC) for different types of autoencoders

Pol et al. [26] proposed a deep learning implementation of anomaly detection using Variational Autoencoders (VAE), also strongly recommended by [40], where his experiments show that VAE and AE outperform Generative Adversarial Networks (GANs) performance in terms of F1 Score, Accuracy, Recall and AUC.

Moreover, the authors of [26] prove that the autoencoder approach can automate the Data Quality Monitoring (DQM) scrutiny, using a tolerance for false negatives. Figure 3.5 illustrates the area under the curve graph (AUC) for different types of autoencoder algorithms, where the variational one has the second highest AUC of 0.9 .

The algorithm adopted by the authors is supposed to solve the problem of the data quality monitoring system, which involves analysing histograms based on a quick first-pass analysis of events seen by the detector. The goal is to provide real-time feedback to detector experts to address issues promptly using autoencoders. These provide robust detection by performing quality assessment and minimising reconstruction errors during training. Further, the autoencoders can identify deviations from the learned standard patterns, signalling potential anomalies. However, fully unsupervised VAEs struggle to achieve perfect disentanglement, leading to multiple latent variables influencing the same observed features. Next, the algorithms disentangle the factors of variation in the dataset by identifying and isolating specific latent factors contributing to the observed data.

The authors also classified anomalies based on two types: *type A Anomalies*, and *type B Anomalies*. The first type involves significant changes only in one feature, while the second is more subtle and involves systematic changes across a structural configuration group (collective behaviour).

The model's architecture also aims to solve the Variational Autoencoders (VAE) disentanglement issue by identifying the known and unknown factors in the representation. The authors propose a model based on a function of known (k) and unknown (u) latent vectors ($x = f(k; u)$), where it is assumed that k and u are marginally independent. The authors then use structural configuration groups, which capture a subset of the k vector and form a structural group. This is aimed at capturing specific patterns or structures in the data.

Training of the autoencoder model is firstly done on the standard (non-anomalous) data. Here, the model learns to encode the input data into a compact representation (encoding) and then decode it to reconstruct the original data while minimising the difference between the input and the reconstructed output.

The last step is detecting anomalies, using *threshold fixing* where the potential anomalous instances are flagged.

3.6 Generative-adversarial networks

Generative-adversarial networks (GANs) are another type of Deep Learning deep learning algorithm, which adds a new component: *an adversary* on top of the generative model. The generative model tests the adversary by producing fake data and using it without detection. In contrast, the adversary model acts as a detective, trying to spot the fake (anomalous) entries generated by the first component. The intuition aims to improve both components' performance until their counterfeits are not distinguishable from genuine articles [12]. Goodfellow further formulates the problem as a min-max game using the following equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.8}$$

where $\min_G$ shows that the generator G minimises the value of the equation, and $\max_D$ shows that discriminator D maximises the result of the equation. $\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)]$ represents the expected value of the log probability that the discriminator correctly identifies real data samples (x) as real, $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$ represents the expected value of the log probability that the discriminator correctly identifies generated data samples G(z) as fake. The value of D(x) is expected to be around one when it takes as input real data X. If the input comes from the generator, the value of D(G(z)) is expected to be around 0 [8]. This means that the better one component gets, the worse the other component becomes, implying an inverse proportionality relationship. Thus, the generator aims to minimise its loss function, defined by the following equation:

$$\max_{V_D} = \sum_{i=1}^{m} \log(D(x_i)) + \sum_{j=1}^{k} \log(1 - D(G(z_j))) \tag{3.9}$$

Where x_i stands for the i-th object in the actual data X, m stands for the total number of objects in X, k is the total number of samples generated by the generator, z_j is the j-th fake data, $D(x)$ is the output of the actual data, and $D(G(z))$ is the probability of correctly identifying the accurate data [8].

In its turn, the discriminator is trying to maximise its success rate at classifying inputs as real or fake, which is defined by the following equation:

$$\min_{V_G} = \sum_{j=1}^{k} \log(1 - D(G(z_j))) \tag{3.10}$$

Where $\log(1 - D(G(z_j)))$ represents the log probability that the discriminator classifies the generated sample as fake. Here, the generator expects its mapped noise to have a similar value as the distribution of the actual data X [8]. In other words, the generator tries to make the discriminator's job as hard as possible by generating indistinguishable samples from accurate data.

According to Langr et al. [15], GANs have several components, depicted in Figure 3.6:

- **Real Data (X)** - also known as the training set, has the instances that the generator G must learn to generate
- **Random Noise Vector (z)** - Raw input to the generator, composed of random numbers.
- **Generator (G)** - Learns the distribution of input data. Generates fake examples ($G(z)$) resembling real data.

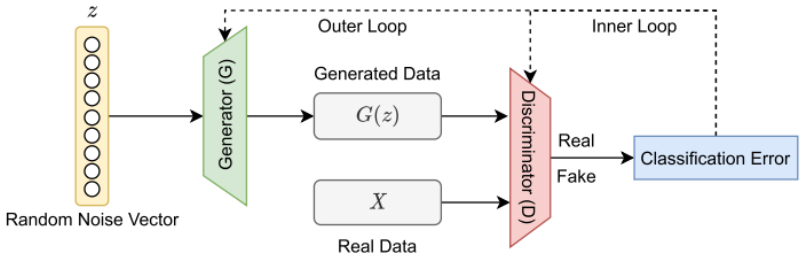

FIGURE 3.6: GANs components, where the classification error is used to update the parameters of the discriminator and generator models - Sabuhi et al. [20]

- **Discriminator (D)** - Distinguishes between real and generated data. Outputs binary decisions (real/fake).
- **Iterative Training** - GANs are trained using the classification error of the discriminator, which is used to tune first the discriminator's parameters, then the generator's parameters. Here, back-propagation is the usual choice as a training algorithm and contains two loops:
  - Inner loop - Tune discriminator parameters to maximise classification accuracy for real and generated data.
  - Outer loop - Tune generator parameters to minimise the chance of detection by the discriminator.

Usually, the Generator (G) is the crucial component in most use cases, as it has to generate fake examples well enough to challenge the Discriminator (D). However, the second component usually stands out in *anomaly detection*. This process happens because the discriminator determines the model's performance in detecting anomalies.

Anomaly Detection using GANs

Xusheng Du et al. [8] propose a GAN-based network structure that includes a generator, a discriminator, and an autoencoder. Their approach is based on an ensemble of the basic structure of a GAN (a generator, a discriminator, and an autoencoder) as depicted in Figure 3.7. The generator is trained to fit the normal object distribution in unsupervised conditions, whereas the autoencoder is trained with fake data generated by the GAN to learn the deep features of standard objects.

Moreover, the authors further use adversarial learning to fit the potential distribution of a dataset in a self-supervised manner, addressing the misclassification problem often encountered in conventional methods due to difficulty fitting the data distribution. In their experiments, the authors show that the GAN-based Unsupervised Outlier Detection (GUOD) method has a better performance when compared to ten baseline algorithms, including k-nearest Neighbours (KNN), Local Outlier Factor (LOF), k-means, isolation forest, and autoencoders.

To detect anomalies, the authors perform several steps in their experiments:

1. **Training the GAN** - this step is done using the gradient descent formula, as illustrated in Figure 3.8, where the authors show how the fake data from random points gradually reaches the distribution range of standard objects. Given a dataset X of standard objects and outliers, *the generator* prioritises fitting the normal object distribution and maps random noise to this distribution, minimising network error. Hence, the generator's fake data can extend the set of standard objects in X.

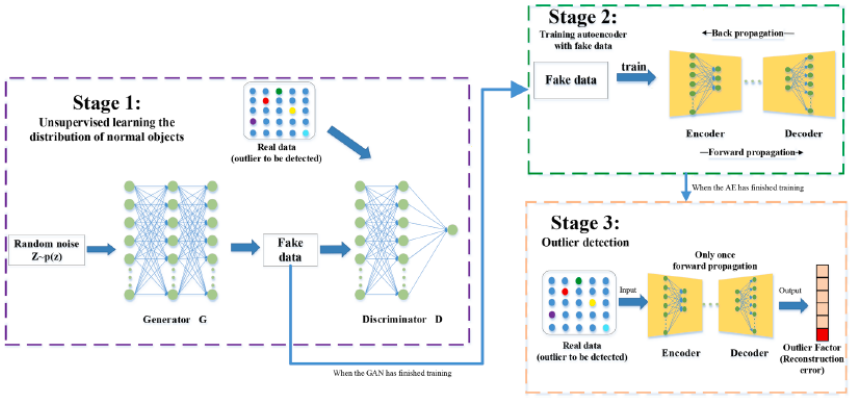

FIGURE 3.7: Architecture of GAN for anomaly detection - Du et al. [8]

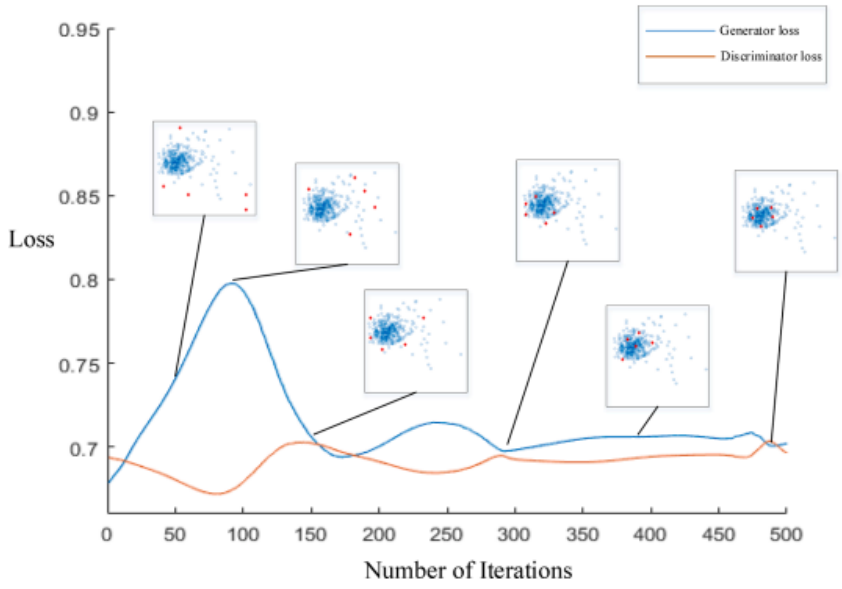

FIGURE 3.8: Changes in distribution of fake data during GAN training [8]

2. **Training the autoncoder** - is the next step, based on augmented data, where both the encoder and decoder use the sigmoid activation function $\sigma(W_1X + b_1)$, while the loss function is the Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.11}$$

where y_i are the actual observed values, $\hat{y}_i$ are the predicted values, and n is the total number of observations or instances. Thus, the ensemble of models operates in two stages. In the first one, the GAN network learns the distribution of normal objects unsupervised, as described at the beginning of this section.

3. **Calculation of the outlier factor** is the last step of the anomaly detection process, where the *outlier factor* represents the degree of deviation of an object from others, where a higher value means a higher likelihood of the instance being an outlier. In their solution, the authors use the reconstruction error of the autoencoder as the outlier factor.
During the second stage of autoencoder training, the dataset to be detected is fed into the autoencoder for one forward propagation. The autoencoder, trained on 'normal' objects, learns their deep features. Therefore, regular objects in the dataset could easily be reconstructed using the autoencoder, while outliers increase the reconstruction error. The output of the ensemble model after autoencoder reconstruction is denoted by $o_{ij} \in O$. The outlier factor is then calculated using the formula:

$$OF_i = \frac{1}{d}\sum_{j=1}^{d}(x_{ij} - o_{ij})^2 \tag{3.12}$$

where OF_i is the outlier factor for the i-th object, o_{ij} denotes x_{ij} after autoencoder reconstruction, and d denotes the dimensionality of x_i.

Finally, in their experiment, this method proved to have the highest performance among the eleven algorithms selected: AutoEncoder, Local Outlier Factor, k-Means, isolation forest and k-Nearest Neighbours. Its performance is measured using the Area Under the Curve (AUC), Receiver Operating Characteristic (ROC), Accuracy (ACC), Detection Rate (DR), False Alarm Rate (FAR), and runtime metrics. *Accuracy (ACC)* measures the proportion of total correct predictions (both true positives and true negatives) out of all predictions, with the formula:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.13}$$

Detection Rate (DR) measures the proportion of true positives out of actual positive cases, with the formula:

$$DR = \frac{TP}{TP + FN} \tag{3.14}$$

False Alarm Rate (FAR) measures the proportion of false positives out of actual negative cases, with the formula:

$$FAR = \frac{FP}{TN + FP} \tag{3.15}$$

Here, TP and TN denote the algorithm's number of correct classifications, and FP and FN indicate the number of outliers the algorithm misclassifies as everyday objects and the number of normal objects that it mistakes as outliers, respectively. Higher values of ACC and DR and lower values of FAR indicate the algorithm's better detection performance.

GAN is not limited only to one implementation since the algorithm has several variations, which are discussed by Sabuhi et al. [20]. The authors mention different variants of GANs used for anomaly detection, where the most significant are the following:

1. **Deep Convolutional GAN (DCGAN)** is a type of GAN that uses convolutional layers in its generator and discriminator networks, unlike a normal GAN, which can use any kind of layer in its generator and discriminator networks. The generator and the discriminator behave identically to the discussion of the paragraph above. The difference between the generated and accurate data can then be used as an anomaly score. This variant uses convolutional and convolutional-transpose layers in the discriminator and generator, avoiding fully connected layers. Here, the discriminator contains stridden convolution layers, batch normalisation layers, and LeakyReLU activations.

2. **Wasserstein GAN (WGAN)** is a type of GAN that improves based on the original GAN's training stability and generated sample quality. It uses a different loss function, the Wasserstein loss, calculated as in Equation 3.16. This loss function measures the distance between the distribution of the generated data and the actual data, making it a suitable choice for anomaly detection. The smaller the Wasserstein distance, the more similar the generated and accurate data are, and vice versa. Thus, the formula for the objective function is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] + \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{3.16}$$

   the first two terms are the original WGAN objective (critic and generator loss). The last term is the gradient penalty multiplied by λ, standing for the penalty coefficient.

3. **Conditional GAN (cGAN)** is a variant of GANs that allows the generation of data conditioned on certain types of information, providing more control over the data generation process. This method can be beneficial in anomaly detection, where the type of anomaly (e.g., fraud, network intrusion) can be used as a condition to generate data. The generated data can then be compared with the actual data to detect anomalies.

   Where $\mathbb{E}_{x\sim p_{data}(x)}[D_X(x) - D_X(F(x))]$ ensures that the discriminator (D_X) can distinguish between authentic images x from domain X and fake images $F(x)$ generated by the generator (F), and $\mathbb{E}_{y\sim p_{data}(y)}[D_Y(y) - D_Y(G(y))]$ this term ensures that the discriminator (D_Y) can distinguish between authentic images (y) from domain Y and fake images ($G(y)$) generated by the generator (G).
   Next, $\lambda \mathbb{E}_{x\sim p_{data}(x)}[||F(G(x)) - x||_1]]$ ensures that if one transforms an image (x) from domain X to domain Y using (G) and then back to domain X using (F), one should get back the original image (x). The parameter (λ) controls the importance of this term, while $\lambda \mathbb{E}_{y\sim p_{data}(y)}[||G(F(y)) - y||_1]$ term ensures that if one transforms an image (y) from domain Y to domain X using (F) and then back to domain Y using (G), one should get back the original image (y).

In conclusion, these algorithms can help detect violations in each data quality dimension discussed in Chapter 2. Any significant deviations in the generated data could indicate, for *completeness*: missing information in some regions of the dataset, and for *accuracy*: significant deviations from the expected output could indicate inaccuracies in the original data.

For *consistency*, it can learn to generate consistent data across different data sources, where significant inconsistencies in the generated data could indicate inconsistencies in the original data. Regarding *uniqueness*, the models can generate unique synthetic customers, where any significant similarities in the generated data can indicate duplication in the original data.

For *validity*, all GANs can be trained to generate valid data that follow specific rules, where any significant deviations in the generated data could indicate invalid entries in the original data. Finally, for *timeliness*, the GANs can be trained on actual data and produce records based on the most recent trends. If the generated data shows significant deviations from the expected output based on the most recent trends, it could indicate outdated entries in the original data.

Concluding the discussion, anomaly detection proves to be a suitable choice in detecting data quality issues. Isolation Forests, Autoencoders and Generative Adversarial Networks (GANs) are three powerful algorithms used for Anomaly Detection in several areas, so comparing their performance can significantly enhance the data quality issue detection from the banks. Thus, the thesis proposes a solution using a parallel implementation of an ensemble model (bagging), which enables the comparison and combination of the rows detected by each method. The results are then validated using the expertise and internal data analysts' results.

Chapter 4

Experiments' preparation

The first part of this chapter discusses the steps corresponding to the fourth step of the DSR: *Designing and Development of the Solution* (in Section 1.5), explained in the methodology section. The chapter further explains the data preparation (manipulation) step, which is necessary before running the experiments. The manipulation phase is divided into data extraction, processing, and analysis substeps.

4.1 Solution's Workflow

This section outlines the methods for anomaly detection in our solution, focusing on identifying data quality issues. The approach is structured into the following key phases: Data Extraction, Data Processing, Data Analysis, Model Development, and Result Validation, represented in a workflow diagram.

Ensemble: Isolation Forest and Autoencoders

To achieve the goals defined in the Design Science (Chapter 1.5), we construct an ensemble solution based on the following artifacts and processes illustrated by Figure 4.1:

- **Isolation Forest**: This algorithm isolates anomalies based on the shortest tree paths. It builds an ensemble of trees for the dataset to compute anomaly scores. The thesis uses the Sci-Kit Learn module to implement Isolation Forest, where the parameters are mostly kept default, except contamination and max_samples, which adapt to the data set they are applied on (see Chapter 5).
- **Autoencoders**: These neural network models are used to learn a compressed data representation. In the thesis, this algorithm is implemented using the Keras module in Python, where seven layers are used to build the model's architecture. Lastly, the reconstruction errors are analyzed to detect anomalies, explained in Chapter 5.
- **Top K-Selection**: The top 20 anomalies are selected from the results of the two algorithms. Next, the typical results of the two algorithms are extracted, where the top 3 unique customers are sent for validation to Subject Matter Expertise (SMEs) from ING.
- **Results explanation**: Applying Shapley Values, an Explainable AI on the top 20 records classified as anomalies by each model, gives insights into the feature contribution to the results detected by the model.

Data sets

The preparation of the experiments starts from the internal credit risk data sets of ING. Gaining access to the credit risk data set inside a bank, especially one of the largest in Europe, was crucial

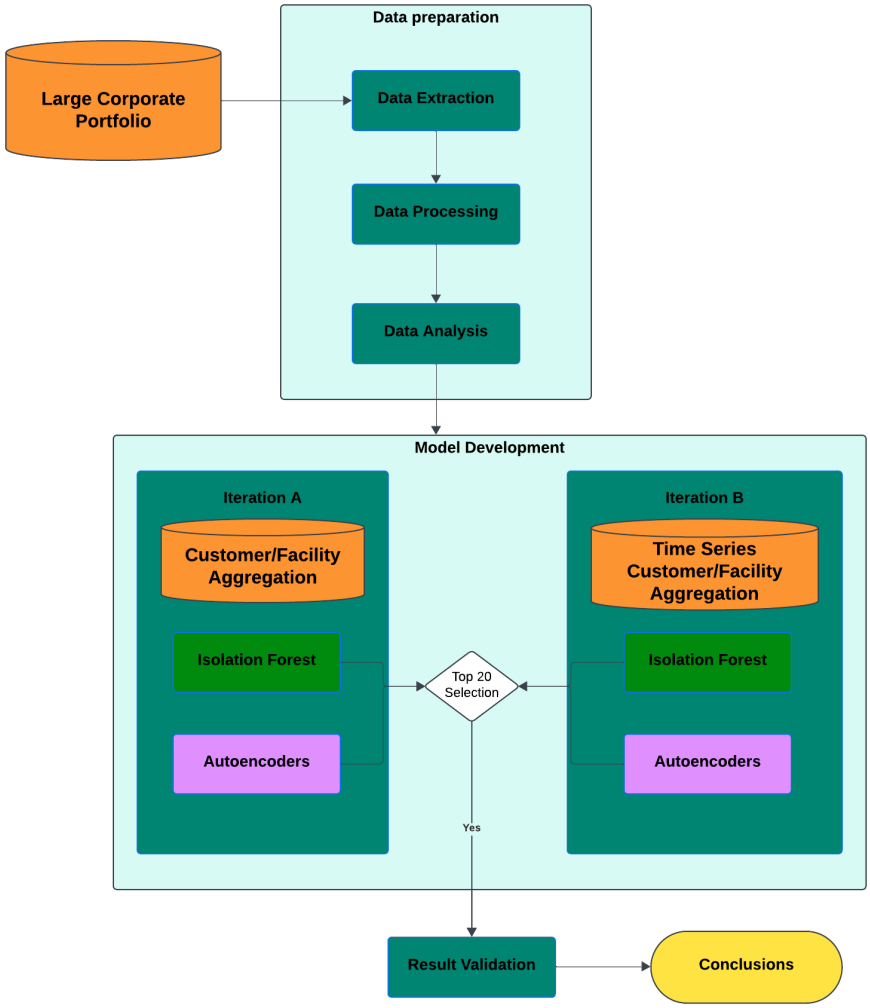

FIGURE 4.1: Approach Implementation

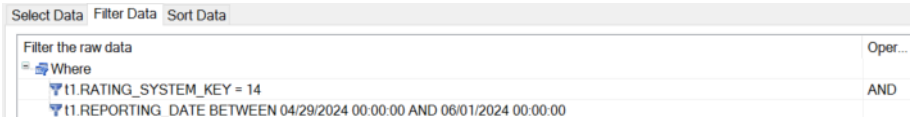

FIGURE 4.2: Filtering on the fact table with RATING_SYSTEM_KEY = 14 selects only the large corporate customers and REPORTING_DATE on April and May 2024

to implement anomaly detection solutions on actual business data. The data sets used in this work are based on the ING's large corporate portfolio. Internally, the portfolio is structured into Internal, External, and Remediated data sources, where the *Internal* data source has been the focus of our experiments, as it contains the original data before remediation. The tables used in this thesis arrive from internal sources, including the central data system, where the credit risk calculations are stored monthly.

Access to these sources has been obtained through the SAS Enterprise Guide, a SAS-based tool, where ING stores its data sets and queries on servers. Since these sources store credit information from corporate customers, key figures are limited to numbers and do not reveal the names/IDs.

The base data set was extracted from a specific data library of ING that stores the historical credit risk metrics. The data set is a fact table containing loan portfolio data of ING Bank for both Business and Consumer exposures aggregated at the outstanding/cover level, with 325 features, retaining a total of 18 months of reporting dates. The focus is here on the *credit risk metrics* discussed in Chapter 3, where the data quality issues can significantly impact the data set. The following attributes were extracted from this table, among which the most notable ones are: *RWA*, *LGD*, *PD*, *RWF (Risk Weighted Factor)*, *EAD*, *EXPOSURE (Amount)*, *MATURITY (Rate)*, *MAX LIMIT (of amounts a customer could take for loan)*, *OS (Outstanding)*, and *PROVISIONS*, shown in Table 4.1.

The other data sets used are situated in other internal databases, merged on top of the extracted one, containing information about Offices, Customers, Limit Purpose, Risk Rating, and Industry Type that contains information where data quality issues can be potentially found.

4.2 Data pre-processing

As ING analysts work with credit risk samples, stored in the lowest level of granularity (outstanding/cover), its data sets require several transformation operations to the highest level (customer/facility level), where only the attributes relevant for the experiments part are stored. This section covers the steps performed in the data processing phase.

Table extraction

First, the relevant attributes are extracted from the central fact table (the credit risk portfolio data set), illustrated in Figure 4.2. Here, the filter is applied for RATING SYSTEM = 14, the corresponding value for the Large Corporate customers. Next, because the amount of data we wanted to send to the experts would be too large, we filtered the data set to include only the last two consecutive reporting dates to generate times series analysis format (Dataset B) for *April*, and *May* as in Figure 4.2.

Merge operations

Next, to gather all the necessary information for the data analysis, we extracted more granular records (Cover, Office, Industry etc.) from the following tables, and left-joined them in this order to the fact table extraction:

Column Name	Source Column
CUSTOMER_ID (CUSTOMER_ID)	t1.CUSTOMER_ID
REPORTING_DATE (REPORTING_DATE)	t1.REPORTING_DATE
RWA_R (RWA_R)	t1.RWA_R
RWF_R (RWF_R)	t1.RWF_R
CUSTOMER_KEY (CUSTOMER_KEY)	t1.CUSTOMER_KEY
EAD_R (EAD_R)	t1.EAD_R
LGD_R (LGD_R)	t1.LGD_R
PROVISIONS_R (PROVISIONS_R)	t1.PROVISIONS_R
EAD_E (EAD_E)	t1.EAD_E
EAD_I (EAD_I)	t1.EAD_I
PD_R (PD_R)	t1.PD_R
ALLOC_LIMIT (ALLOC_LIMIT)	t1.ALLOC_LIMIT
EAD_P (EAD_P)	t1.EAD_P
OS_R (OS_R)	t1.OS_R
OS_E (OS_E)	t1.OS_E
EXPOSURE_E (EXPOSURE_E)	t1.EXPOSURE_E
MAX_LIMIT (MAX_LIMIT)	t1.MAX_LIMIT
OUTSTANDING_ON_R (OUTSTANDING_ON_R)	t1.OUTSTANDING_ON_R
NET_SALES (NET_SALES)	t1.NET_SALES
PAST_DUE_AMOUNT (PAST_DUE_AMOUNT)	t1.PAST_DUE_AMOUNT
MATURITY_R (MATURITY_R)	t1.MATURITY_R
MIS_RAROC_PRODUCT_KEY (MIS_RAROC_PRODUCT_KEY)	t1.MIS_RAROC_PRODUCT_KEY
CUSTOMER_TYPE_KEY (CUSTOMER_TYPE_KEY)	t1.CUSTOMER_TYPE_KEY
INITIATING_OFFICE_KEY (INITIATING_OFFICE_KEY)	t1.INITIATING_OFFICE_KEY
MATURITY_R1 (MATURITY_R1)	t1.MATURITY_R1
COUNTRY_INCORP_KEY (COUNTRY_INCORP_KEY)	t1.COUNTRY_INCORP_KEY
DEPARTMENT_KEY (DEPARTMENT_KEY)	t1.DEPARTMENT_KEY
EXPOSURE_CLASS_OFFICIAL_KEY (EXPOSURE_CLASS_OFFICIAL_KEY)	t1.EXPOSURE_CLASS_OFFICIAL_KEY
FACILITY_ID (FACILITY_ID)	t1.FACILITY_ID
LIM_INIT_BASE_ENTITY_KEY (LIM_INIT_BASE_ENTITY_KEY)	t1.LIM_INIT_BASE_ENTITY_KEY
LIM_BOOK_BASE_ENTITY_KEY (LIM_BOOK_BASE_ENTITY_KEY)	t1.LIM_BOOK_BASE_ENTITY_KEY

TABLE 4.1: Table from the Credit Risk Dataset Columns and Their Sources

- OFFICE_BASE_ENTITY - getting information about the office responsible for managing the loan
- EXPOSURE_CLASS - getting information about the financial status of the Large Corporate customers
- INDUSTRY_TYPE - information regarding the industry where the customer is currently active
- RISK_RATING - the rating system and score for a customer
- RISK_RATING_MODEL - a tool used to assess the probability of default for a borrower or a specific credit exposure. These models use various factors to generate a numerical or symbol-based rating that summarizes the level of default risk.
- RATING_SYSTEM - measure and differentiate credit risk across individual credits and

groups of credits. It monitors changes and trends in risk levels, improving the risk management process.

- LEGAL_ENTITY - contains information about the organization or company that is legally recognized as a separate entity from its owners.
- PRODUCT_TYPE - contains information related to the type of facility (product) lent by a customer.
- CUSTOMER_SEGMENT - information on the customers' categorization based on specific criteria such as industry, size, revenue, geographic location, or risk profile.
- COVER_TYPE - information of the type of collateral or security provided to mitigate the risk of default.
- LIMIT_PURPOSE - information about the specific reason or objective for which a credit limit is set.

Aggregation

Once merged, all the clients with only **one** reporting date were removed from our scope to have an accurate time-series representation of the data set. This step was necessary to calculate the inter-monthly change rates, which can be impossible for the customers having only one reporting date in the data set. Finally, for the modelling, two data sets were obtained:

1. **Dataset A** - aggregated on the `CUSTOMER_ID/REPORTING_DATE/FACILITY_ID` level due to granularity reasons, containing information about each Facility of a Customer on a Reporting Date, which further contributes to the analysis of the 10 Credit Risk Metrics from the LC Portfolio.
2. **Dataset B** - aggregated on the `CUSTOMER_ID` level, consists of the inter-monthly ratios between April and May of the 10 Credit Risk Metrics analyzed.

Normalization

The last step following aggregation is normalizing the data to enhance the model's performance by ensuring a consistent data range. This result has been achieved by applying the MinMaxScaler from the SciKitLearn library to the numerical values of the aggregated dataset, which scales all features to a specified range (between 0 and 1). This normalization process is crucial for optimizing the model's accuracy and efficiency.

4.3 Data Analysis

This section describes in detail the data analysis step, which is crucial for understanding the data before performing manipulation operations for the model development part.

4.3.1 Aggregation level

The data analysis was conducted on the original aggregation level of the credit risk data set (also the lowest - *outstanding/cover level*), which provided insights of the data before aggregating it to the format used in the modelling part - the *customer/facility level*, described in the section above.

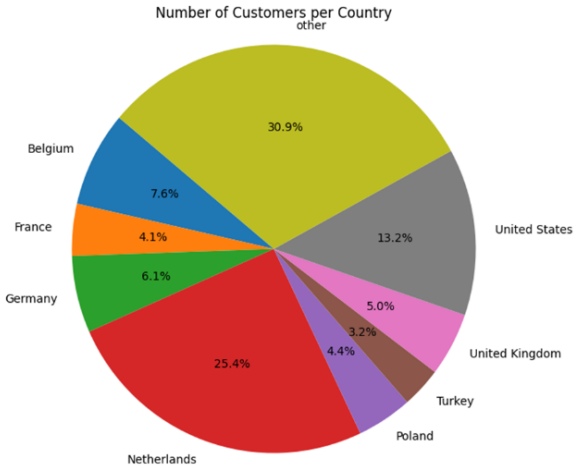

FIGURE 4.3: Number of customers per country

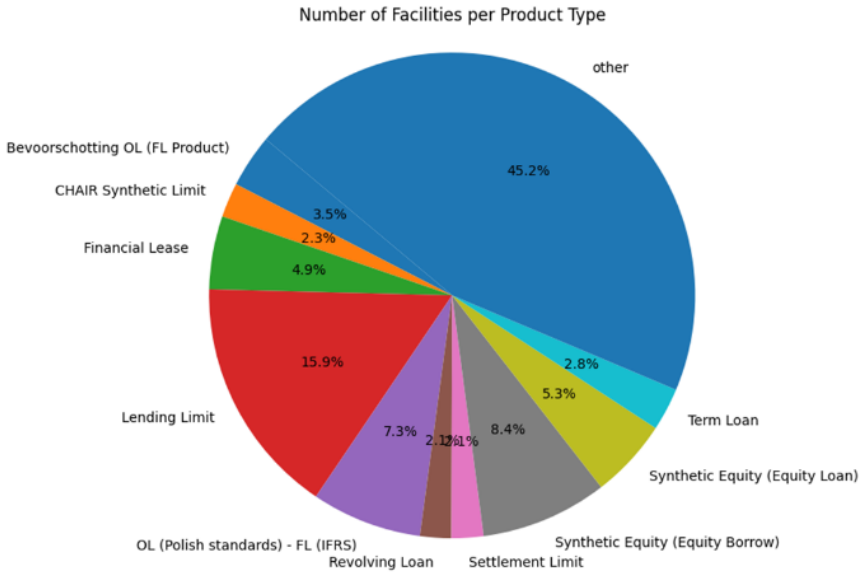

FIGURE 4.4: Number of Facilities per Product Type

4.3.2 General insights

The analyzed data set has *13,254* unique customers, sampled on two reporting dates: *April 30, 2024*, and *May 31, 2024* to be able to provide a time-series analysis representation of the data set, and to allow receiving validation results on time.

Customers

The most significant number of Large Corporate customers is from the Netherlands, as depicted by Figure 4.3.

Facilities

There are 99,058 unique facilities in the Large Corporate Portfolio, where among the product types with the most facilities is the lending limit, depicted in Figure 4.4.

4.3.3 Dataset A - Outstanding/Cover level (not-aggregated)

Below, we perform a complete statistical analysis of the first data set (Dataset A).

Descriptive Analysis

We considered all the credit risk metrics mentioned above, looking for the count, mean, standard deviation, minimum value, maximum value, median and percentiles, as depicted in Table 4.2.

TABLE 4.2: Statistical Analysis of the data set (non-aggregated)

Metric	OS	RWA	EAD	LGD	PD	RWF	MATURITY	EXPOSURE	MAX LIMIT
count	466,192	466,192	466,192	466,192	466,192	466,192	466,192	244,460	466,192
mean	1,307,769	257,979	1,273,663	0.322	0.040	0.628	1.915	1,785,105	31,346,476
std	25,726,614	3,004,181	25,218,543	0.220	0.131	0.757	1.395	13,998,229	98,051,507
min	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000
25%	0.000	0.000	0.000	0.112	0.004	0.110	0.400	307	11,781
50%	1,807	241	1,380	0.432	0.005	0.357	1.906	12,337	1,155,367
75%	37,467	5,516,474	24,507	0.436	0.015	0.884	2.918	94,461	16,400,000
99%	22,700,198	6,087,516	23,635,176	0.709	1	3.365	5	40,932,301	644,834,416
max	8,365	14,008	677,464,994	4,008	1.000	9.571	5.000	1,122,281	2,600,000

The descriptive analysis of the financial ratios highlights significant variability and *skewness* across the dataset. Key metrics such as OS, RWA, EAD, and others exhibit mean values around zero, indicating a central tendency, while standard deviations and interquartile ranges (IQR) demonstrate substantial variability.

The presence of extreme values at the 99th percentile suggests significant outliers, which is critical for understanding the risk and quality of the data. Skewness in the data distributions indicates asymmetry, which could impact the accuracy of predictive modelling.

Concluding the descriptive analysis, it was necessary to apply normalization techniques before developing the model, as described in Section 4.2.

Duplicates

Next, a check for duplicates is performed, where no duplicates can be found.

Missing values

Concerning *missing data values*, EAD_P, OUTSTANDING_ON_R, and PAST_DUE_AMOUNT were the columns with the highest percentage of missing values, depicted in Figure 4.5.

Outliers

A scatter plot of the Mahalanobis distances visualizes the data set's distribution. This plot helps to identify a natural separation between normal data points and potential outliers.

For outlier detection in our data set, we selected the Mahalanobis distance method for its effectiveness in handling multivariate data, considering the correlations between variables to measure the distance of each data point from the mean.

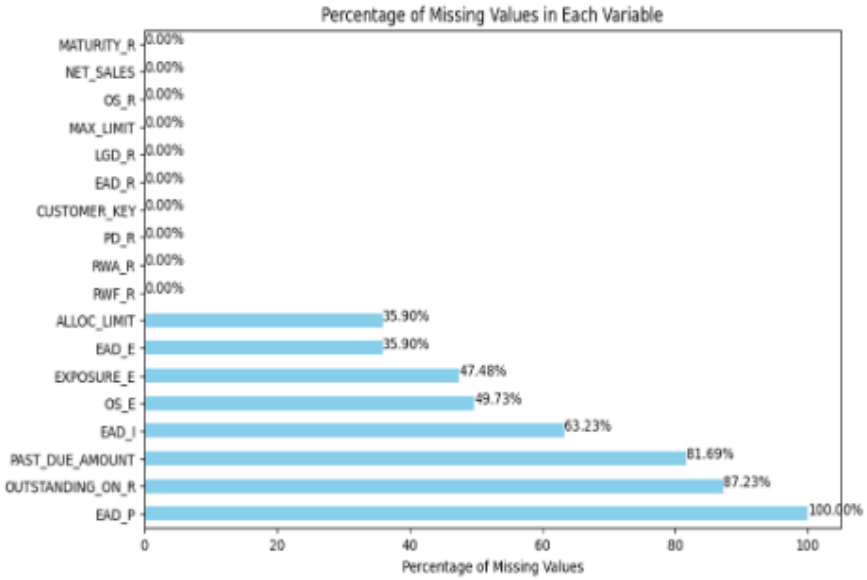

FIGURE 4.5: Missing values of Dataset A

Threshold setting

The subsequent critical step involved determining the optimal threshold. Using a cross-validation-based approach, the optimal threshold was identified to be around the 94^{th} percentile. However, we set the threshold at the 99^{th} percentile to target the most extreme outliers in the data set.

Correlation. Variable Dependence

The correlation analysis revealed direct proportional relationships between key financial variables. The highest correlations were observed between Exposure at Default (EAD) and both Outstanding (OS) and Exposure Amount (EXPOSURE). These relationships indicate that as EAD increases, OS and EXPOSURE tend to increase proportionally. This is visually represented in the heatmap of Figure 4.7, highlighting significant interdependencies crucial for understanding the dynamics within the dataset.

FIGURE 4.6: Scatter Plot of Feature Distribution - outliers highlighted in red

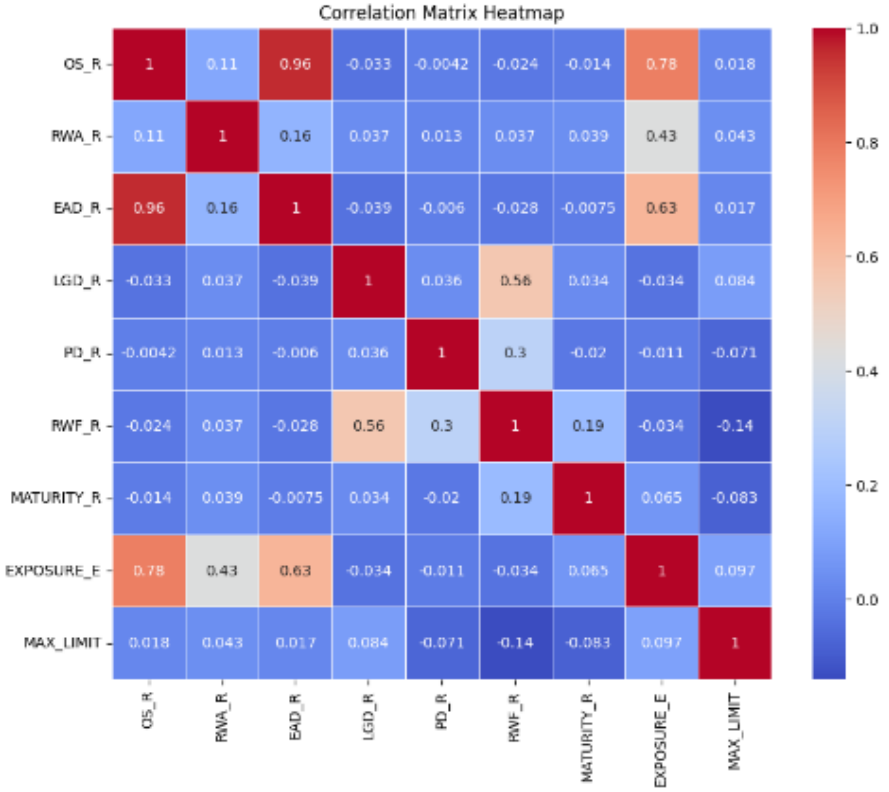

FIGURE 4.7: Correlation Matrix of the Dataset A (non-aggregated)

4.3.4 Exceptional case analysis

We identified an unexpected pattern during our exploratory data analysis for the non-aggregated dataset (Dataset A). This pattern represents a few customers with an Exposure at Default (EAD) value of 0, while their Probability of Default (PD), Loss Given Default (LGD), and Risk-Weighted Assets (RWF) metrics were positive. Domain experts confirmed this unexpected behaviour and indicated anomalies within the dataset.

Following the advice of these experts, we decided to implement a data-cleansing strategy. Specifically, we removed customers with more than one record where PD, LGD, or RWF metrics were positive, while their EAD was 0. However, we retained customers with only one record with positive PD, LGD, or RWF metrics with an EAD of 0.

This filtering process revealed that approximately *0.017%* of the data set's records (80 rows) exhibited this anomalous property.

Removing these obvious outliers resulted in the model's focus on extreme outliers.

4.3.5 Dataset B - Inter-monthly log ratios data set

The next phase of the analysis focuses on examining the inter-monthly variations for two consecutive months for the ten credit risk metrics. This specific data format allows us to closely monitor abrupt changes between consecutive reporting dates for each customer, facilitating the detection of potential anomalies.

By tracking these sudden shifts, we can identify irregular patterns that could indicate underlying issues or risks. This approach provides a valuable tool for early warning and proactive credit risk management.

To transform the raw data set (Dataset B) into this format, we first aggregated it into the *customer/facility level*, where we then used feature engineering to create logarithmic ratios for each of the ten credit risk metrics. We calculated the logarithmic ratios of each numerical value using the following formula:

$$\text{Metric}_{Ratio} = log(\frac{Metric_{May}}{Metric_{April}}) \tag{4.1}$$

Where $\text{Metric}_{\text{April}}$ stands for the value of the respective credit risk metric with 30^{th} April 2024 as the reporting date, while $\text{Metric}_{\text{May}}$ represents the value of the respective credit risk metric on 31^{st} May 2024. The aggregation here was changed to the *facility/reporting date level*, which allows us to perform time-series analysis on the data set for the two consecutive months.

Descriptive analysis

As for the raw data set, we performed statistical analysis on the time-series data set (inter-monthly ratios), shown in Table 4.3.

The descriptive analysis of inter-monthly credit risk ratios reveals key insights about the data and potential anomalies, similar to Dataset A. The mean values for these ratios are close to zero, indicating minimal average monthly changes. However, the substantial standard deviations and wide ranges point to significant variability. For instance, the RWA Ratio, with a mean of -0.034 and a standard deviation of 0.378, spans from -17.935 to 19.722, while the EAD Ratio has a mean of -0.022 and a standard deviation of 0.313, ranging from -18.195 to 15.454. This wide variability suggests the presence of notable outliers or anomalies in the dataset.

Outliers

Following the same approach as for the raw data using *Mahalanobis distance* with the threshold set for 99%, we calculated the percentage of outliers for each column, as Figure 4.9 shows:

TABLE 4.3: Statistical Analysis of the inter-monthly rates (log ratios)

Metric	RWA RATIO	EAD RATIO	PD RATIO	LGD RATIO	OS RATIO	MAX LIMIT RATIO	EXPOSUR RATIO	MATURITY RATIO
count	70,392	70,392	70,392	70,392	70,392	70,392	70,392	70,392
mean	-0.034	-0.022	0	-0.008	-0.023	-0.021	-0.024	-0.013
std	0.378	0.313	0.190	0.194	0.416	0.243	0.350	0.209
min	-17.935	-18.195	-5.691	-5.049	-18.018	-17.744	-18.018	-6.411
1%	-0.850	-0.612	-0.531	-0.367	-1.042	-0.400	-0.693	-0.172
25%	-0.027	-0.015	0	0	-0.017	-0.019	-0.015	-0.031
50%	0	0	0	0	0	0	0	0
75%	0	0	0	0	0	0	0	0
99%	0.558	0.366	0.587	0.052	0.868	0.107	0.418	0.162
max	19.722	15.454	7.289	4.897	15.454	12.391	11.185	6.411

Correlation. Ratio dependence

Then, we looked for correlations between logarithmic ratios finding only direct proportionality relationships. The highest correlations are this time between the Exposure at Default (EAD) and the Risk Weighted Assets (RWA) ratios, followed by Exposure amount and Outstanding ratios, illustrated in Figure 4.9:

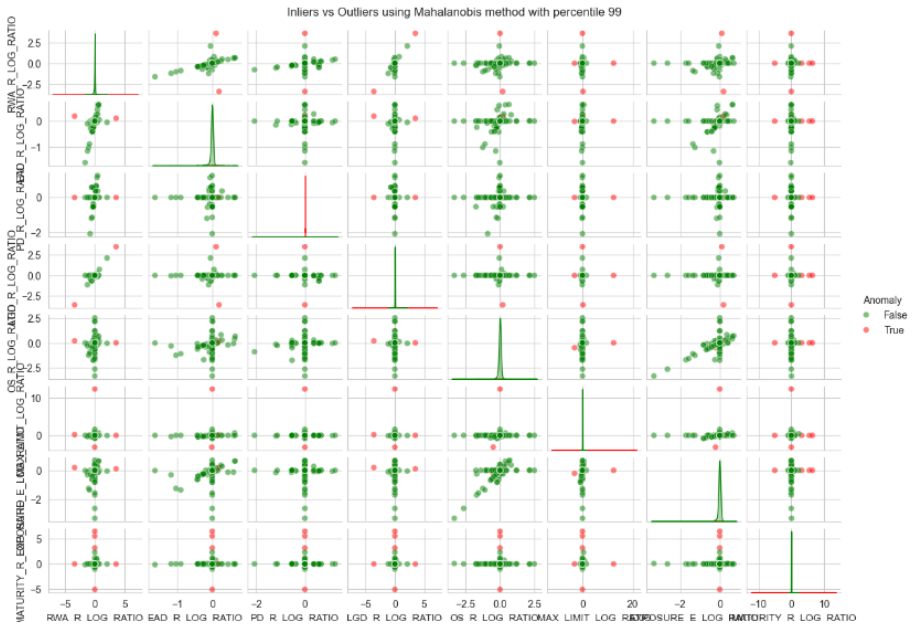

FIGURE 4.8: Scatter plot of outlier distribution for Dataset B

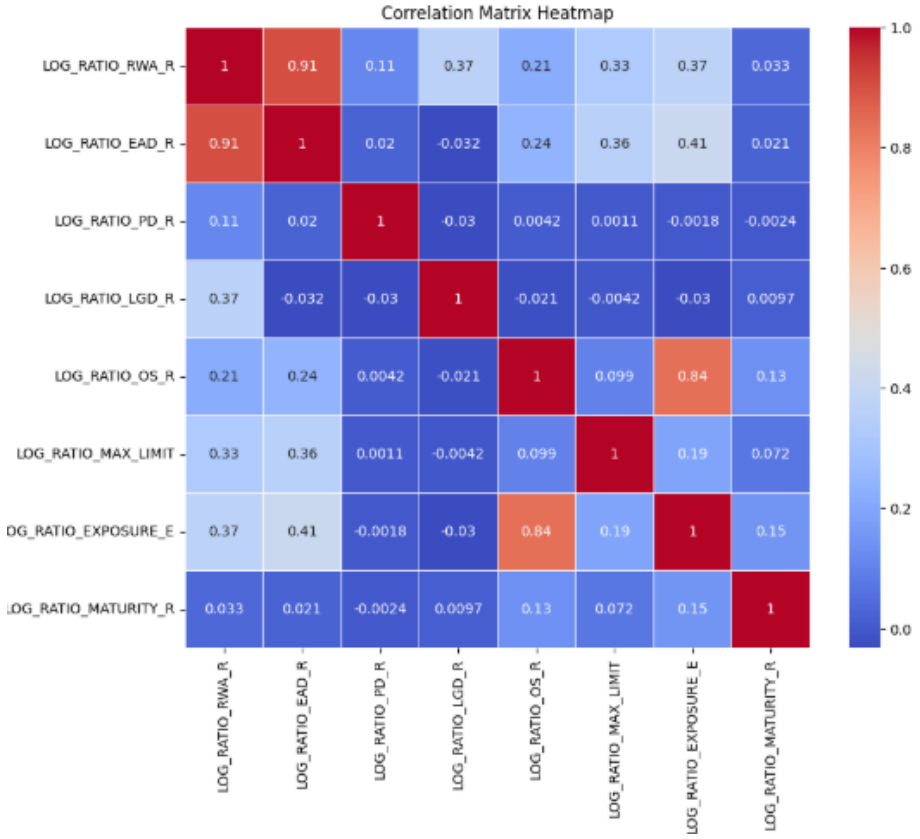

FIGURE 4.9: Dataset B - Correlation Matrix for the Inter-monthly rates

Chapter 5

Model development

This chapter explains the architecture and the development reasoning of the two models, Isolation Forest and Autoencoders, and continues with a detailed overview of the experiments performed.

5.1 Ensemble modelling

This phase is split into two experiments with different combinations of the researched methods:

- Iteration A - Isolation Forest and Autoencoders on the aggregated data set
- Iteration B - Isolation Forest and Autoencoders on the log ratios data set

Each experiment in terms of iterations was conducted using *bagging (parallel)* ensemble learning type, where the results are calculated as the commonly detected rows and sorted by the average anomaly probability.

As the data set was mainly skewed, before running the model, we applied *Min-Max Scaling* for Dataset A on the numerical features while ensuring there is no missing value or duplicate in the data set. Next, even though splitting is not mandatory, we applied an 80/20 training/test split to evaluate the model's performance on unseen data, given its unsupervised nature.

5.1.1 Train, test and validation strategies

An important step for the modelling part is the data split. The Isolation Forest model was trained on the entire dataset to maximize the learning potential from all available data using the `fit` method. Given the unsupervised nature of anomaly detection, this approach allows the model to comprehensively understand both normal patterns and anomalies within the dataset. Utilizing the entire dataset ensures the model is exposed to the full range of data variations, essential for enhancing its ability to detect subtle anomalies.

Next, the same *threshold* of 99^{th} percentile is used as a threshold for the anomaly detection (called with the `predict` method from the `sci-kit learn` library) - see Figure 5.2.

```
def train_isolation_forest(data, contamination=0.01, n_estimators=1000,
random_state=42, max_samples=0.5):
    iso_forest = IsolationForest(n_estimators=n_estimators,
     contamination=contamination, random_state=random_state,
                                  max_samples=max_samples)
    iso_forest.fit(data)
    return iso_forest
```

FIGURE 5.1: Using "fit" method to train Isolation Forest

```python
def predict_anomalies(iso_forest, data):
    anomaly_scores = iso_forest.decision_function(data)
    predictions = iso_forest.predict(data)
    data['Anomaly score'] = anomaly_scores
    data['Anomaly'] = predictions
    top_anomalies = data.sort_values(by='Anomaly score', ascending=True).head(20)
    anomaly_scores = np.sort(anomaly_scores)
    shap_values = calculate_shap_values(iso_forest, top_anomalies.iloc[:, :-2])
    return data, anomaly_scores, predictions, shap_values, top_anomalies
```

FIGURE 5.2: Isolation Forest Predict method to detect anomalies

```python
def select_non_anomalous_rows_ae(df, percentile=99,
                                 is_log_ratios=False):
    df = prepare_data_non_anomalous(df)
    mask = apply_mahalanobis(df, percentile)
    normal_instances = df[mask]
    plot_results_non_anomalous_ae(df, mask, percentile, is_log_ratios)
    return normal_instances
```

FIGURE 5.3: Function for selecting the non-anomalies using Mahalanobis distance

Autoencoders and Threshold setting

On the other hand, a more complex strategy has been implemented for the auto-encoders, motivated by its Deep Learning nature. Firstly, the model is trained solely on data classified as non-anomaly due to its approach that reconstructs the data into an original format [42]. Here, the *threshold* is set using the same method as for data analysis - the statistical method of *Mahalanobis Distance*. This choice is motivated by the model's core engine, which reconstructs the data to its original format. Since the model is trained on normal data, it struggles more to reconstruct anomalies when tested on our dataset (with anomalies). An anomaly is identified as a data point above the threshold the autoencoder model determines, having a higher reconstruction error. The 95/5 training/validation split has been applied for the data set, where the training is set to the default of 100 epochs using shuffling to ensure a good generalization. The next parameter, batch size, is set to 32, a popular choice in the research literature [18].

The detection step is performed on the entire data set, acting as the test set. This approach ensures that the model learns the normal data (non-anomalies) distribution effectively, improving its ability to detect deviations (anomalies) during testing. It comprehensively evaluates the model's performance, including its generalization capabilities and robustness in different scenarios. Careful consideration of performance metrics and validation techniques can mitigate the overfitting risk, ensuring a robust and accurate model.

5.1.2 Models architecture

This subsection consists of the chosen architecture of the model, with the necessary explanations. The parameters presented in this section are the final version which produced the best results, while other versions are discussed in the experiments part, in Chapter 5.

Isolation Forest

For running *Isolation Forest*, we considered the following parameters [32] :

- **Contamination:** 0.01 - Useful when the proportion of anomalies in the dataset is unknown, allowing the model to adapt to the data distribution.

- **N_estimators:** 100 - Specifies the number of base estimators in the ensemble. The default parameter provides a balance between computational efficiency and model performance.
- **Max_samples:** 0.5 - Common amongst researchers, trains each base estimator on a random subset containing 50% of the training data. This value is a good trade-off between bias and variance and helps to reduce overfitting.
- **Random_state:** 42 - Controls the randomness of the bootstrapping of the samples and the feature selection. It allows for consistent results across multiple runs.
- **Bootstrap:** False - Default value, if True - individual trees are fit on random subsets of the training data sampled with replacement. If False - sampling without replacement is performed.

Autoencoders

For *Auto-encoders*, we considered the following architectural design with one input and output layer, one encoder and decoder, and one "bottleneck layer" as illustrated in Figure 3.4: Next, the chosen hyper-parameters are the following:

- **GlorotUniform Initializer**: `GlorotUniform(seed=42)` - Ensures weights are initialized to values that promote faster convergence and better training stability.
- **BatchNormalization**: `BatchNormalization()` - Normalizes input layer for faster training and improved performance.
- **Input Layer**: 10 (input dimension) - Defines the input data shape necessary for building the model.
- **Dense Layers (Encoder)**:
  - `Dense(12, activation="relu", kernel_regularizer=l1_l2(l1=0.02, l2=0.02), kernel_initializer=initializer)` - Uses ReLU activation, regularization to prevent overfitting, and proper weight initialization.
  - `Dense(8, activation="relu", kernel_regularizer=l1_l2(l1=0.02, l2=0.02), kernel_initializer=initializer)` - Further reduces dimensionality with same motivations.
  - `Dense(4, activation="relu", kernel_regularizer=l1_l2(l1=0.02, l2=0.02), kernel_initializer=initializer)` - Bottleneck layer with same motivations.
- **Dense Layers (Decoder)**:
  - `Dense(8, activation='relu', kernel_regularizer=l1_l2(l1=0.02, l2=0.02), kernel_initializer=initializer)` - Symmetrical to encoder for consistent training.
  - `Dense(12, activation='relu', kernel_regularizer=l1_l2(l1=0.02, l2=0.02), kernel_initializer=initializer)` - Similar structure to encoder layers.
  - `Dense(input_dim, activation='sigmoid', kernel_initializer=initializer)` - Final layer to reconstruct input data.
- **Optimizer**: `Adam(learning_rate=0.00015)` - Adam, short for Adaptive Moment Estimation, is an optimizer that adapts the learning rate for each parameter. It combines the benefits of two other popular optimization methods, AdaGrad and RMSProp, to efficiently handle sparse gradients and non-stationary objectives. The learning rate, set to

0.00015, controls the step size during each iteration of the optimization process, determining how much to adjust the model weights in response to the estimated error each time the model weights are updated.

- **Loss Function**: `Mean Squared Error (MSE)` - Measures error between input and reconstructed output to guide optimization. Calculated with the formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  Where:

  - n is the number of observations
  - y_i is the actual value
  - $\hat{y}_i$ is the predicted value

- **ReduceLROnPlateau Callback**: `ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=0.000015)` - Reduces learning rate on validation loss plateau.
- **EarlyStopping Callback**: `EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)` - Stops training early if validation loss doesn't improve.
- **Random Seed**: `np.random.seed(42); tf.random.set_seed(42)` - Ensures reproducibility of results.
- **Training Data Preparation**: `prepare_data_ae(data, test_size=0.05, random_state=42)` - Splits the data into training and validation sets, which is essential for model evaluation.
- **Training Procedure**: `train_autoencoder(train_data, val_data, loss_function)` - Defines the training process, including the optimizer, loss function, and callbacks for reducing learning rate and early stopping.

5.1.3 Explainable AI

On top of each detected anomaly by both models, *Shapley Values* - an implementation of Explainable AI has been applied to enable the ability to understand the reasoning behind each model for detecting a specific row as an anomaly. Using Shapley Values, the percentage of each feature contributing to the detection of a specific row has been calculated.

However, the calculation differs from both models, as *Isolation Forest* uses *anomaly score* parameter to indicate whether a point is an anomaly with a range between -1 (anomaly) and 1 (normal), contrary to the Autoencoders. The latter method uses reconstruction error, which ranges between 0 (normal) and 1 (anomaly), leading to the following interpretation regarding the Shapley Values:

1. Lower values - higher contribution in the detection using Isolation Forest
2. Higher values - higher contribution in the detection using Autoencoders

5.1.4 Result interpretation

The chosen architecture consisted of a parallel ensemble (bagging) to check models' consensus in detecting anomalies and potential *data quality issues*. For clarity in the interpretation, the *top 20 anomalies* are extracted separately for each model, together with their calculated Shapley Values.

Next, the common results between the two models are stored in a DataFrame, where both *anomaly score* and *reconstruction error* are scaled using MinMax to probabilistic values, adding two extra columns altogether in the common results data set. Table 6.5 from Chapter 5 illustrate

5.1.5 Validation

To provide a solid validation of the models used, we applied two types of validation on top of the results of both Isolation Forest and Autoencoders

1. **Quantitative validation** – using formal methods and metrics to validate model performance.

2. **Qualitative validation** – using subjective judgement and domain expertise to interpret the model's results (if the record is an anomaly or not).

Quantitative Validation

The evaluation methods used in our experiments are:

1. For Isolation Forest:

   - **Distribution test:** Histogram distribution of anomaly scores, ranging from –1 to 1.
   - **Robustness test** by checking whether the detected records are the same after five consecutive runs.

2. For Autoencoders:

   - **Distribution test:** Reconstruction Error Histogram for the training
   - **Loss tests**: Learning Curve with Training and Validation Loss.
   - **Robustness test** by checking whether the detected records are the same after five consecutive runs.

Qualitative Validation

The qualitative validation described in this subsection is performed using two approaches:

1. Validation against an internal dataset from ING contains exceptions detected already for a specific period, including both *random* and *systemic data quality issues*. The success criteria for the models for this step is determined by the percentage of the three metrics: Precision, Recall and F1 Score.

2. Validation against the Subject Matter Expertise (SMEs), specialists reporting daily data quality issues found. As their time is limited, the validation has been limited to the top 3 customers from the data set containing the common records of anomalies detected by Isolation Forest and Autoencoders. The success criteria here are less strict, as the SMEs can only detect *systemic* data quality issues using the false positive rate.

Thus, these diverse methods ensure that both models are evaluated from different perspectives.

5.2 Experiments

This section presents the experiments done inside ING Bank Netherlands using its internal customer data set, which contains over 500,000 records. The experiments are done over two data sets:

1. **Dataset A'** - Dataset A aggregated to the Customer/Facility level filtered, containing customer information from two reporting dates.

2. **Dataset B'** - Dataset B, where only the columns containing the log ratios are kept to prevent noise.

5.2.1 Iteration A (Dataset A')

The first iteration consists of several experiments on Dataset A', illustrated by Table 5.1.

In our study, there were a total of 70 experiments conducted (solely on Dataset A'), where most aimed to tune and stabilize the Autoencoder model. As mentioned in Chapter 5, the initial results sent for validation to the SMEs are produced by an unstable version of the models mainly caused by Auto-encoders, which required several runs to be stabilized. From the beginning of the experiments, Isolation Forest turned out to be stable by using the same parameters mentioned in Section 5.1.2; thus, for simplicity, Table 5.1 only included the Autoencoders relevant parameters that helped stabilize the model.

In the first 17 runs, it can be seen that with a 7-neuron bottleneck layer architecture, the model produced different results for consecutive runs. No callbacks or regularization techniques were used here.

However, when changing the architecture to a more symmetrical one with *four* neurons in the bottleneck layer and introducing *BatchNormalization*, a *GlorotInitializer*, callbacks (*EarlyStopping* and *ReduceLROnPlateau* with patience values) and regularizers after each training epoch, the model performance stabilized. This is illustrated by Table 5.1, where similar results can be seen in consecutive runs for the experiments labelled between numbers 66 and 70.

BatchNormalization has been applied before the training process, which normalizes the activations of the input layer for each subset of the training set, using a random seed of value 42. Given a mini-batch, it computes the mean and variance of activations, subtracts the mean and divides each value by the standard deviation. Among the callbacks, *EarlyStopping* stops the entire training process if the loss does not modify after several consecutive epochs set as a parameter, known as a "patience" value corresponding to the column name "Patience2". Our experiments set the patience rates to 2, improving the model's performance. The last callback used is *ReduceLROnPlateau*, a learning rate scheduler that dynamically adjusts the learning rate after each epoch based on the model's performance on the validation set. This addition helps the model converge uniformly by adjusting the learning rates in training based on a patience parameter, which does not stop the training process, unlike *EarlyStopping*. ReduceLROnPlateau reduces the learning rate if the loss is not decreasing significantly after a certain number of epochs, represented by the column named "Patience1".

Isolation Forest

Firstly, Isolation Forest proved robust initially, constantly detecting the same records. Figures 5.4 and 5.5 show the histogram distribution of the *anomaly scores* for the initial and final versions of the model.

The first graph displays the anomaly score distribution from the initial version of the Isolation Forest model. This histogram reveals that most data points cluster around low anomaly scores, suggesting that the model deems most instances normal. The pronounced peak near the score

Table 5.1: Experimental Results

#	AE Architecture	Output Activation	Hidden Activation	Regularization	Learning Rate	Patience1	Patience2	Common IF-AE	Identical Rows AE
1	16-13-4-13-16	ReLu	ReLu/Tanh	None	0.001	None	3	4	None
2	16-13-4-13-16	ReLu	ReLu/Tanh	None	0.001	None	3	8	10
3	16-13-4-13-16	ReLu	ReLu/Tanh	None	0.001	None	3	9	10
4	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	3	14
5	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	10	14
6	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	11	14
7	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	1	14
17	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	12	14
18	16-14-7-14-16	ReLu	ReLu	None	0.001	None	3	16	14
35	10-12-8-4-8-16	Sigmoid	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	19	18
60	10-12-8-4-8-16	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
61	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
62	10-12-8-4-8-16	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
63	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
64	10-12-8-4-8-16	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
65	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
66	10-12-8-4-8-16	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
67	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
68	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
69	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20
70	10-12-8-4-8-10	ReLu	ReLu/Tanh	L1/L2 (0.02 all)	0.0002	2	2	16	20

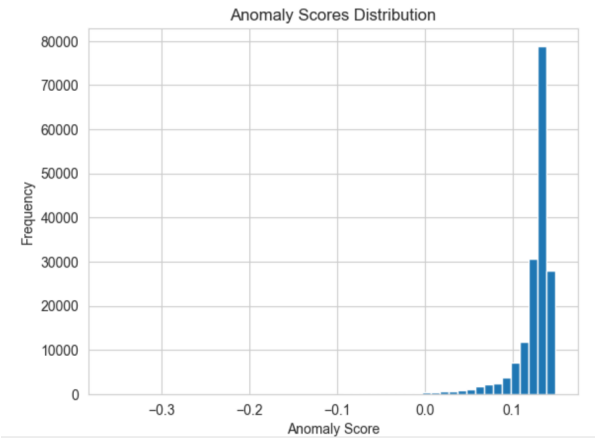

FIGURE 5.4: Histogram distribution of the initial version of Isolation Forest on Dataset A'

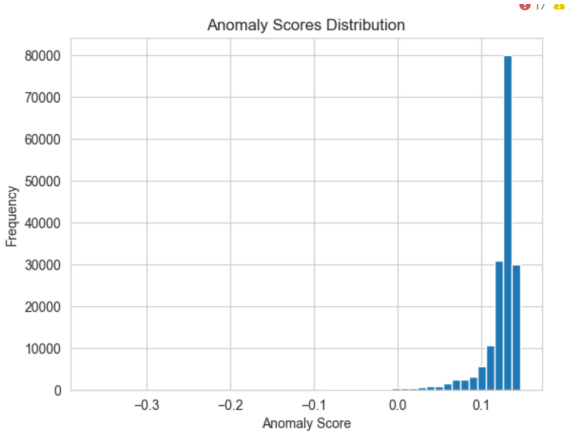

FIGURE 5.5: Histogram distribution of the final version of Isolation Forest on Dataset A'

of 0.1 means that the model consistently identifies a substantial portion of the data as non-anomalous. This initial analysis serves as a baseline for evaluating the model's performance detecting anomalies within the dataset.

The second graph depicts the anomaly score distribution from the final, stabilized version of the Isolation Forest model. Identically to the initial version, most data points continue to cluster around low anomaly scores, with a peak close to 0.1. This consistency in anomaly score distribution before and after stabilization underscores the model's robustness and stability.

Autoencoders

Next, for the autoencoders model, the *quantitative validation* starts with the same histogram method and includes loss tests before and after stabilizing the model.

The first method of the *quantitative validation* for the autoencoder model includes the histograms of the *reconstruction errors* before and after stabilization.

FIGURE 5.6: Reconstruction error histogram of the initial (non-stabilized) version of the Autoencoders on Dataset A'

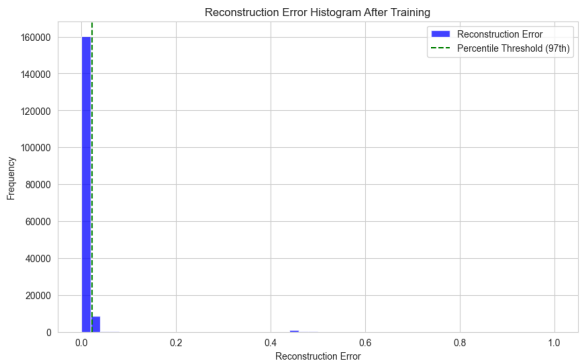

FIGURE 5.7: Reconstruction error Histogram of the final (stabilized) version of the Autoencoders on Dataset A'

It can be seen in Figures 5.6 and 5.7 that even after stabilizing the autoencoder, the *reconstruction error* histogram shows minimal changes, the majority of reconstruction errors remaining concentrated at the lower end of the spectrum. In both the histograms, the reconstruction error ranges from 0 to 1, where most observations are concentrated near zero. The distribution is

heavily right-skewed, with the highest frequencies observed for reconstruction errors close to zero, indicating that the autoencoder effectively reconstructs most normal data points with minimal error. A sharp drop in frequency is observed for higher reconstruction errors, with only a few outliers exhibiting errors significantly above the 97th percentile threshold, marked as a vertical dashed line. This threshold is a cutoff for distinguishing between normal data points and potential anomalies. The results demonstrate the autoencoder's capacity to generalize well to normal patterns in the data while assigning higher reconstruction errors to anomalous observations, which deviate from the learned representation. This consistency suggests that the stabilization techniques improved the model's training and validation losses but did not significantly impact the overall distribution of reconstruction errors. This could be caused by the data's inherent characteristics of the initial model, such as correlation and relationships between the variables (OS with EAD, EXPOSURE and RWF with LGD, EXPOSURE).

Next, for the loss test, the graph in Figure 5.8 shows the instability of the initial autoencoder model, characterized by a sharp spike in validation loss, which signifies poor generalization and overfitting to the training data.

In contrast, the right graph from Figure 5.9 displays the final (stabilized) version, where both training and validation losses decrease sharply and converge closely, indicating enhanced model stability and generalization when applied to unseen data. This stabilization is crucial for ensuring reliable and accurate anomaly detection in practical applications.

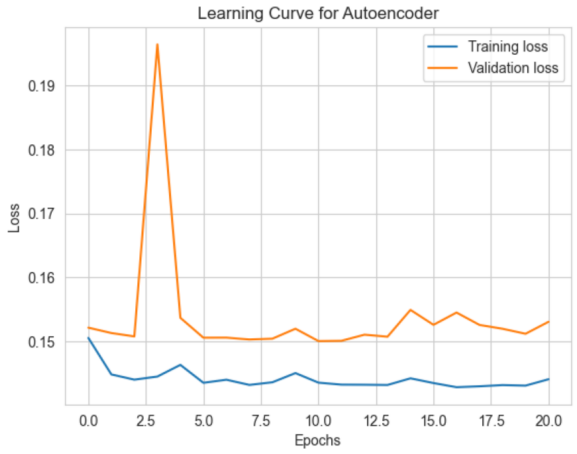

FIGURE 5.8: Learning curve of the initial (non-stabilized) version of the Autoencoders

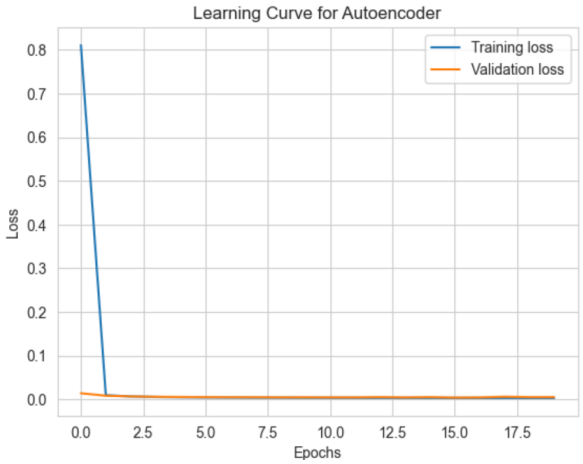

FIGURE 5.9: Learning curve of the final (stabilized) version of the Autoencoders

Despite model stability and generalization enhancements, the autoencoder reconstructs most data points with minimal error, highlighting its efficiency and robustness in the anomaly detection task.

Explainable AI - Shapley Values

The results of both Isolation Forest and Autoencoders (Top 20 Anomalies) are illustrated in Table and the corresponding Shapley Values.

Table 5.2 illustrates the explainable AI results detected by the Isolation Forest model for each feature in the top 3 customers. Here, the principle goes as follows: the lower the value, the higher the column's contribution is to the algorithm's detection of the respective row. PROVISIONS and RWA, with Shapley Values of -0.04 and -0.03, contributed the most to the detection for the first record.

As for the overall contribution of the results for the auto-encoders, Figure 5.10 illustrates the features that contributed the most across the entire Dataset A', where LGD, RWF and PD

TABLE 5.2: Shapley Values for the top 20 anomalies detected by IF

Index	LGD	PD	RWF	OS	MAX_LIMIT	PROVISIONS	EAD	RWA	MATURITY	EXPOSURE
156874	0.00	-0.02	0.00	0.02	-0.00	-0.04	0.01	-0.03	0.00	-0.00
53459	-0.00	0.02	-0.02	0.02	0.00	-0.02	0.01	-0.04	-0.00	-0.00
156873	0.00	-0.02	0.00	-0.07	-0.00	-0.04	0.01	-0.02	0.00	-0.00
43870	-0.00	0.02	0.01	-0.07	0.01	0.03	-0.07	0.02	0.00	0.02
43871	-0.00	0.02	0.01	0.03	0.01	-0.02	0.01	-0.05	-0.00	0.00
53460	0.00	-0.02	0.00	0.03	0.00	-0.06	0.02	0.02	-0.01	0.01
52731	0.00	-0.01	-0.00	0.00	-0.03	0.03	0.03	0.02	0.00	-0.06

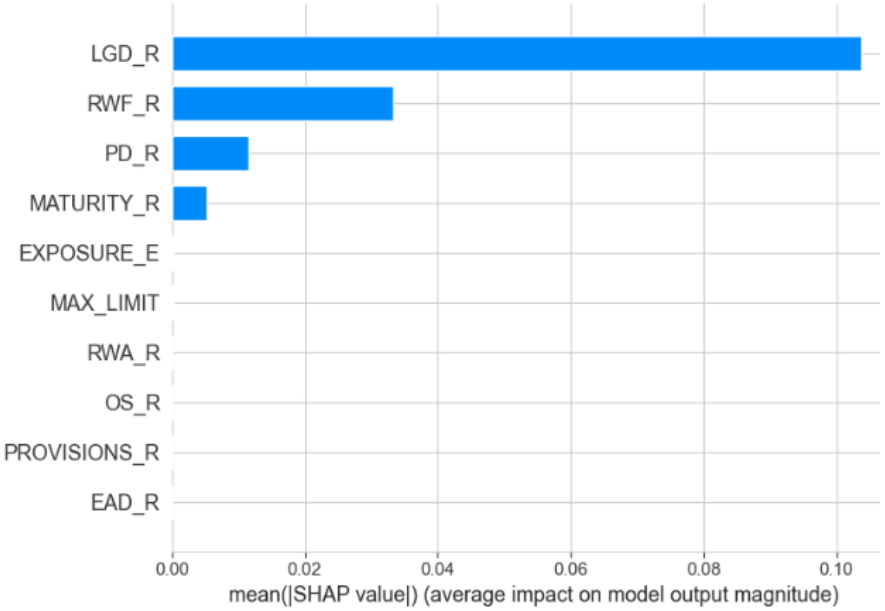

FIGURE 5.10: Average feature contribution for the top 20 anomalies detected by Autoencoders on Dataset A'

are the top 3 contributors. The Shapley value is calculated by applying a mean function on the shape value of one feature for the entire data set.

Results for validation

The last step was to send the results to the validation experts inside ING in an Excel format, illustrated by Table 6.5. The results are the records corresponding to the Top 3 Customers from the common results (detected by both Isolation Forest and Autoencoder).

5.2.2 Iteration B (Dataset B')

For the second iteration, we applied the same steps on Dataset B, which contains the time-series version of Dataset A'. Since the autoencoder model has already been stabilized during Iteration A, the second iteration only consists of 10 experiments, where the models produced the same results, as shown in Table 5.3

Isolation Forest

For Isolation Forest, the histogram distribution illustrated by Figure 5.11 was the same in each of the 10 experiments, showing the robustness of the model.

TABLE 5.3: Experiments on Dataset B'

No.	Test set Separator	Architecture	ID	Batch	Output	Hidden	Loss	Regularization	Learning Rate	Reduce	Patience	Common IF AE	Identical Previous Rows
1	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
2	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
3	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
4	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
5	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
6	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
7	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
8	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
9	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20
10	Mahalanobis/99%	10-12-8-4-8-12-10	100	32	Sigmoid	ReLU	MSE	L1/L2 (0.02 all)	0.00015	5	4	15	20

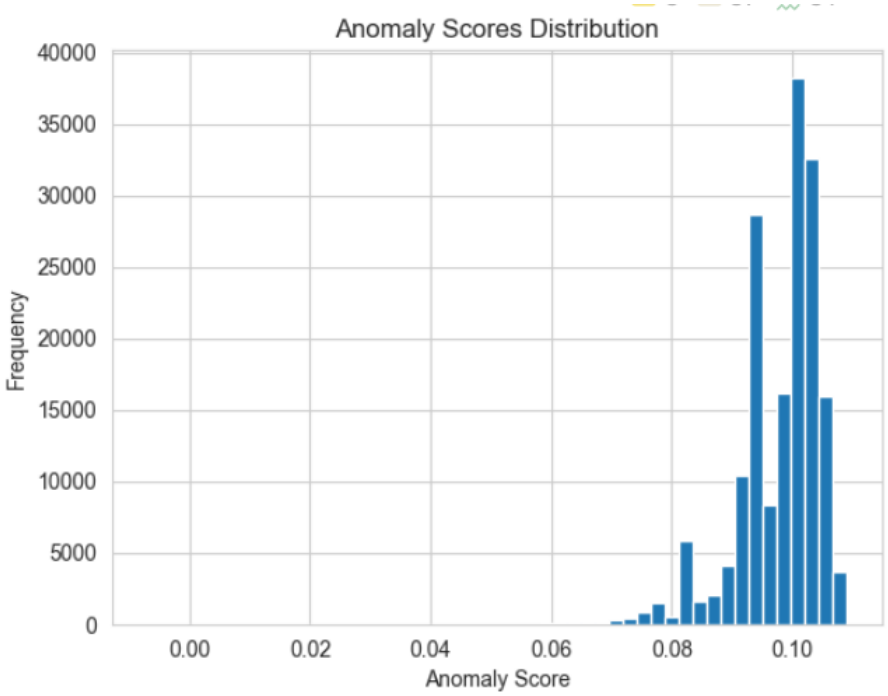

FIGURE 5.11: Histogram of anomaly scores for Isolation Forest on the Dataset B'

The anomaly scores, ranging from approximately 0 to 0.12, exhibit a distinct unimodal distribution with a sharp peak near 0.10. This representation indicates that most data points in the dataset are concentrated within a narrow band of scores, representing normal behaviour. The frequency decreases sharply for anomaly scores deviating from this peak, particularly below 0.08 and above 0.11, suggesting fewer data points in these regions. The highest observed frequency exceeds 35,000, underscoring the algorithm's ability to group most normal observations into a specific score range.

Autoencoders

Regarding Autoencoders, for the *quantitative evaluation*, the reconstruction error plot showed a normal distribution with a sharp peak around 0.01. This illutration means a high percentage of the data points in the set are concentrated on normal behaviour. Illustrated by Figure 5.12, the graph shows a sudden decrease in the frequency between 0,0 and 0.01. The frequency of records with a reconstruction error above the threshold of the 99th percentile is extremely low, with only dozens being detected as anomalies. This graph showcases the algorithm's ability to detect most normal observations in a certain range, separated by the anomalies using a threshold.

The second quantitative method was the loss test using a learning curve graph, illustrated

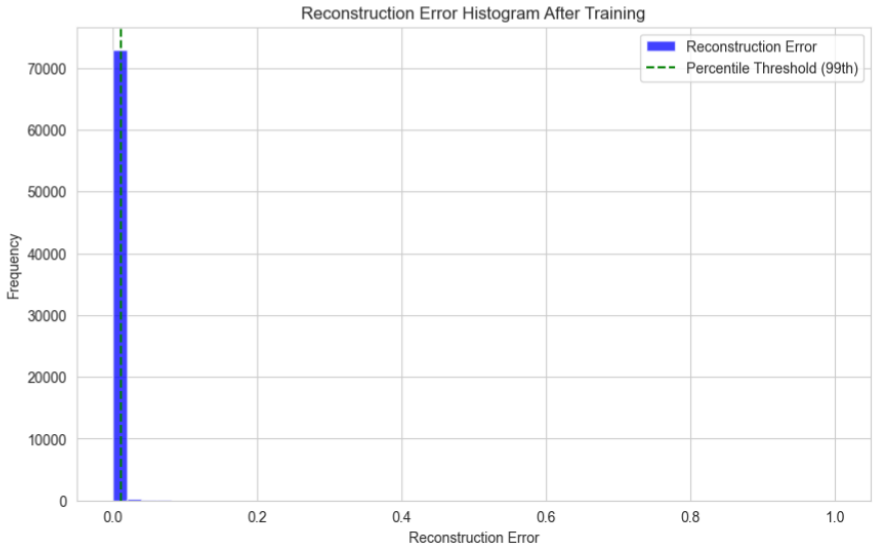

FIGURE 5.12: Reconstruction error histogram of the Autoencoders model on Dataset B'

in Figure 5.13. The plot shows variation in training and validation loss of Iteration B in the 100 training epochs. In the illustration below, it can be seen that both losses converge early, with the validation loss converging slightly faster and at a barely higher value than the training loss in most of the epochs. This shows stability and robustness, indicating that the model improves its performance throughout the training process.

Explainable AI - Shapley Values

For Dataset B', both models produced the results of the features' shapley values for each of the top 20 anomaly rows detected. Table 5.4 illustrates the Shapley Values for the Isolation Forest model, where the lower values represent a higher contribution of the feature to the record being detected as an anomaly by the model.

TABLE 5.4: Shapley Values of the Isolation Forest on the output of Dataset B'

Index	RWA_LR_shap	RWF_LR_shap	EAD_LR_shap	PD_LR_shap	LGD_LR_shap	OS_LR_shap	MAX_LIMIT_LR_shap	EXPOSURE_LR_shap	MATURITY_LR_shap	PROVISIONS_LR_shap
40595	-0.024	0.009	-0.024	0.002	0.006	-0.049	-0.032	-0.023	0.000	0.014
3724	-0.051	-0.016	-0.028	0.002	-0.015	-0.042	0.010	0.025	0.000	0.015
6785	-0.003	0.009	-0.012	0.002	0.005	-0.018	-0.021	-0.008	0.000	0.016
24245	0.012	-0.019	-0.006	0.002	-0.016	-0.011	0.011	-0.003	0.000	0.001
6816	0.014	-0.020	-0.002	0.002	-0.015	-0.006	-0.017	-0.001	0.000	0.017
52287	-0.038	0.014	-0.030	0.002	0.012	0.024	0.010	-0.033	0.000	0.016

Next, the Shapley values for the entire data set are expressed in Figure 5.14, where PROVISIONS, OS and RWA are the features with the highest contribution. Their value is calculated as the average of the Shapley Values of a feature throughout the rows of the model's output.

Results for Validation

The time-series log ratios data set (Dataset B') has not been validated due to time-related issues. This limitation should be considered when interpreting the results.

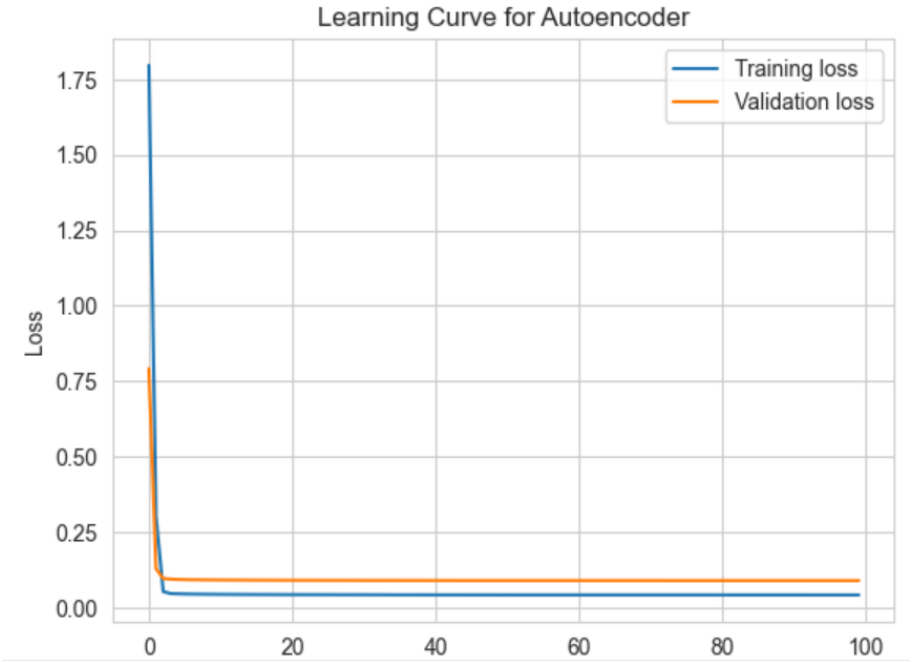

FIGURE 5.13: Learning curve for the Autoencoders on Dataset B'

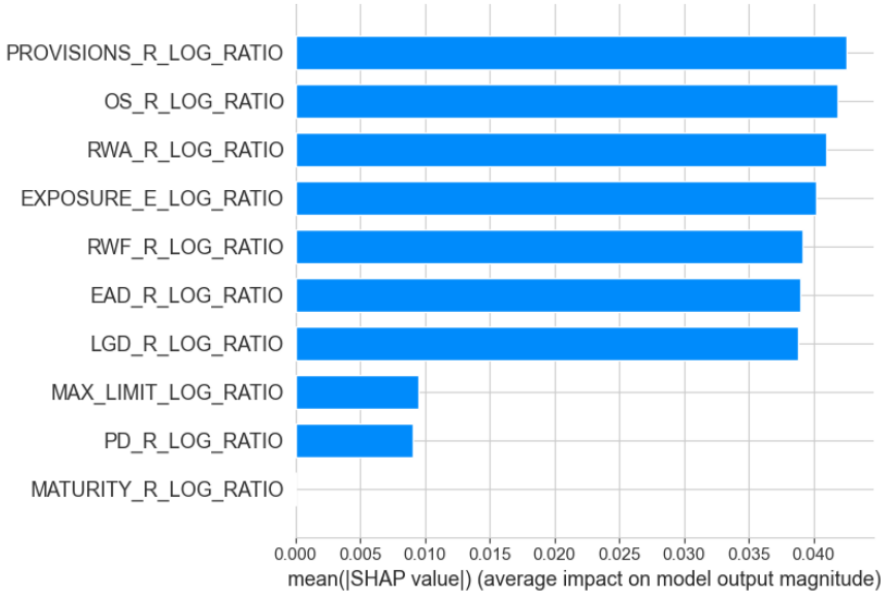

FIGURE 5.14: Average feature contribution for the top 20 anomalies detected by Autoencoders on Dataset B'

Chapter 6

Validation

This chapter corresponds with the sixth phase of the Design Science methodology (Evaluation and Validation). We outline the validation strategy, describe how the results are formatted, and present two methods used for validation: internal data validation using ING's data sets and expert validation through Subject Matter Experts (SMEs). The use of these two methods is needed to assess models' performance in detecting both systemic and random data quality issues.

6.1 Validation Strategy

We validate the model using both expert and non-expert input, leveraging internal data sets that contain both systemic and random data quality issues. The internal data validation (non-expert) focuses on assessing the model's ability to detect potential data quality problems. In contrast, expert validation is dedicated to evaluating systemic data quality issues, allowing for a more precise understanding of the model's effectiveness in identifying genuine data problems. The performance of both methods is aggregated using the arithmetic mean.

6.2 Results' formatting

The results of Iteration A consisted of the top 20 anomaly rows detected by Isolation Forest and Autoencoders, together with a dataset resulted from the common records from the top 20 anomalies detected by each algorithm.

Table 6.1 illustrates these common rows, where besides the original columns, it contains the anomaly probability calculated by each algorithm and the average probability, calculated as a simple arithmetic mean between the probabilities of Isolation Forest and Autoencoder. Here, the *Anomaly Probability IF* and *Anomaly Probability AE* columns are calculated by applying a Min-MaxScaling transformation of the Anomaly score and Reconstruction error, respectively.

Evaluation Metrics

To evaluate the model's performance, we use the following metrics:

- **Precision**: The proportion of detected anomalies that are actual anomalies [29].
- **Recall**: The proportion of actual anomalies that the model correctly identifies [29].
- **F1-Score**: The harmonic mean of precision and recall, offering a balanced measure of the model's effectiveness [35].

We chose only these metrics because of the limitations of the data set, caused by being only able to obtain the True Positives expressed as correctly detected anomalies.

TABLE 6.1: Common anomaly records detected by Isolation Forest and Autoencoders (Iteration A)

Index	Anomaly Probability AE	Anomaly Probability IF	Average Probability
156874	0.8995	1	**0.9498**
43870	1	0.8728	**0.9364**
156873	0.8827	0.9589	**0.9208**
43871	0.9663	0.8539	**0.9101**
53459	0.6477	0.9845	**0.8161**
52731	0.9411	0.4849	**0.7130**
152127	0.6043	0.3591	**0.4817**
152128	0.5867	0.3333	**0.4600**
52730	0.6686	0.2209	**0.4447**
158138	0.6224	0.1118	**0.3671**
158139	0.6224	0.0962	**0.3593**

Results evaluation

Being an unsupervised solution, we did not rely on labels for the results. However, after further inquiring inside ING, we found the data set where the Model Validation team stores the detected exceptions, consisting of both random and systemic data quality issues, which means a limited perspective of evaluation, as we cannot see whether the records were *systemic* issues or not. As can be seen, Table 6.2 illustrates all the rows detected by both our algorithms that were found inside the internal data set of ING. The entirety of the rows could be found in the data set, giving a 100% precision, recall and F1 Score in detecting potential data quality issues.

Thus, the evaluation of the results has two methods. The first concerns potential (systemic and random) data quality issues, where both Autoencoders and Isolation Forest performed 100% in precision, recall and F1-score, shown in Table 6.3.

6.3 Internal Datasets

For *Iteration A*, Table 6.2 illustrates a relevant section of the internal data set of ING with systemic and random data quality issues. As can be observed, all the records detected by Isolation Forest and Autoencoders are present in the internal table.

TABLE 6.2: Internal results - systemic and random data quality issues for Iteration A

Index	LGD	PD	RWF	OS	MAX_LIMIT	PROVISIONS	EAD	RWA	MATURITY	EXPOSURE
53459	0.64622511	0.275	3.98010682	187046995.56	187046995.56	41587586.1	187046995.56	744467023.141	2.62498093	187046995.56
43870	0.001	0.00153432	0.00062104	23013940811.91	0.0	0.0	23013940811.91	14292642.203	1.0	0.0
53460	0.64629412	0.275	3.96887773	168882133.48	168882133.48	36588976.21	168882133.48	670272538.919	2.54286066	168882133.48
152127	0.43571616	0.16323687	2.26722291	1122281973.4	1122281973.4	0.0	0.0	0.0	1.0	1122281973.4
152128	0.43571616	0.16323687	2.26722291	1105430427.0	1105430427.0	0.0	0.0	0.0	1.0	1105430427.0
80845	0.6104557	0.275	3.55374118	104746317.51	104746317.51	16355915.21	104746317.51	372241301.487	1.08772958	104746317.51
80846	0.61047317	0.275	3.54247911	103173506.52	103173506.52	15460765.31	103173506.52	365489991.266	1.00295674	103173506.52
158139	0.35890377	1.0	0.0	123919810.26	122871336.12	55587276.78	123919810.26	0.0	1.83974763	123919810.26
62828	0.4848407	1.0	4.16018135	60602251.53	60602251.53	10136914.86	60602251.53	252116356.78	2.54880681	60602251.53
161222	0.001	0.00233709	0.00081981	11328906261.72	0.0	0.0	11328906261.72	9287589.872	1.0	0.0
94527	0.730648	1.0	9.5714888	8731349.21	10000000.0	0.0	727597.27	6964189.121	1.0	8731349.21
18682	0.4881906	1.0	6.39529686	0.0	1.0	0.0	0.1	0.6395	5.0	0.0
18690	0.4881906	1.0	6.39529686	0.0	1.0	0.0	0.1	0.6395	5.0	0.0
18685	0.4881906	1.0	6.39529686	0.0	1.0	0.0	0.1	0.6395	5.0	0.0

The performance score of this method is calculated as the average of precision, recall, and F1 score, with a total score of 100%, averaging the results from Table 6.3.

For *Iteration B*, Table 6.4, similarly as for Iteration A, illustrates the unique rows detected by both Autoencoders and Isolation Forest as Top 20 anomalies found in the internal data set of ING, containing random and systemic data quality issues.

TABLE 6.3: Model Performance Metrics for the First Validation on Iteration A

Model	Precision	Recall	F1 Score
Isolation Forest	100%	100%	100%
Auto-encoders	100%	100%	100%

TABLE 6.4: Internal data quality issues - systemic and random data quality issues for Iteration B

Index	RWA_LR	RWF_LR	EAD_LR	PD_LR	LGD_LR	OS_LR	MAX_LIMIT_LR	EXPOSURE_LR	MATURITY_LR	PROVISIONS_LR
40595	-17.54	0.48	-18.02	0	0.48	-18.02	-17.74	-18.02	0	0
3724	19.72	4.27	15.45	0	4.27	15.45	0	0	0	0
6785	-10.86	0.61	-11.48	0	0.61	-11.48	-11.48	-11.48	0	0
24245	-6.07	3.57	-9.64	0	3.58	-9.64	0	-9.64	0	-4.07
6816	-5.22	3.54	-8.77	0	3.54	-8.77	-8.77	-8.77	0	0
52287	14.86	-0.01	14.86	0	0	0	0	14.86	0	0
49028	-11.10	-2.76	-8.33	0	-2.76	-8.33	-8.33	-8.33	0	0
10858	-17.94	0.26	-18.20	0.42	-0.02	-8.81	-0.02	-8.81	0	0
60481	-9.79	0.01	-9.80	0.34	0	-9.80	0	-9.80	0	-9.51
58384	-8.65	-2.02	-6.62	0	-2.02	-6.62	0	-6.62	0	-10.90
32095	-4.29	3.57	-7.86	0	3.58	-7.86	0	-7.86	0	-2.36
39110	1.91	-0.01	1.92	0	-0.01	11.18	0	11.18	0	10.76
21782	1.79	0	1.79	0	0	9.97	0	9.97	0	9.88
10895	8.66	-0.01	8.67	0	0	0	8.67	8.67	0	0
7589	-9.05	0	-9.05	0	0	-0.33	-0.02	-9.74	0	-10.32
71344	5.78	5.75	0.04	3.60	4.29	0.08	0	0	0	9.02
34326	-4.95	-3.73	-1.21	0	-3.73	-7.71	0	-7.71	0	0
32603	-9.19	-0.27	-8.92	-0.53	0	0	-8.92	-8.92	0	0
34408	5.82	4.23	1.59	0	4.23	1.59	1.59	1.59	0	7.22
32461	-8.78	-0.27	-8.52	-0.53	0	0	-8.52	-8.52	0	0
43948	-12.23	-0.99	-11.25	0	-0.99	-0.46	-0.46	-0.46	0	-3.62
36200	0	0	0	0	0	0	-16.52	0	0	0
35876	-7.29	0	-7.29	0	0	-7.29	-7.29	-7.29	0	0
25517	0	0	-0.02	0	0	-0.02	-16.09	-0.02	0.22	0.10
25516	0	0	-0.02	0	0.19	-0.02	-16.03	-0.02	0.16	0.03

6.4 Subject Matter Expertise (SME)

The second evaluation consists of Subject Matter Expertise (SME) answers for the top 3 customers, corresponding to the first six rows of the common results from the non-stabilized version of the model from Table 6.5. The first two rows, corresponding to the indexes *156873* and *156874* correspond to the first customer.

TABLE 6.5: Initial Results sent to the SMEs

Index	LGD	PD	RWF	OS	MAX_LIM	PROVISIONS	EAD	RWA	MATURITY	EXPOSURE
156873	0.3943	1	1.299	2.61E+08	2.48E+08	76977010	2.61E+08	3.39E+08	1	260821951
156874	0.3971	1	1.422	2.64E+08	2.48E+08	76081468	2.64E+08	3.75E+08	1	263697304
52730	0.4840	1	0	1.24E+08	1.24E+08	65946362	1.24E+08	0	0.247	124000000
52731	0.4840	1	0	1.24E+08	1.24E+08	98108901	1.24E+08	0	0.162	124000000
16376	0.0014	0.000683	0.000491	1.44E+10	1.44E+10	0	1.44E+10	7080076	1	0
16377	0.001	0.001009	0.000464	1.4E+10	0	0	1.4E+10	6495068	1	0

The *expertise's response* stated that one out of three customers was indeed a *systemic data quality issue*, presented in Table 6.6.

To evaluate the performance in this step, we use a stricter evaluation, which implies calculating the percentage of the rows confirmed by the experts as *systemic data quality issues*. Table 6.5 shows that only two of six records were confirmed as *systemic data quality issues*, giving a performance score of 33%.

The final score is calculated using a weighted average of the two scores: 100% for potential data quality issues and 33.3% for actual (systemic) data quality issues. The final result of 66.5% shows a relative imbalance in the ensemble solution, where both models perform excellently in detecting potential data quality issues. However, in terms of real (systemic) data quality issues, the ensemble shows a much lower performance of 33.3%, indicating limitations in relying solely on AI models without expertise, a fact discussed more in detail in Chapter 7.

TABLE 6.6: Feedback received from the SMEs for the top 3 customers detected by both models

Expert Responses

- **Customer 1** - "Don't see an issue here. It is a defaulted customer, i.e. PD is 100%. In these cases, RWA can be higher than limitoutstanding. Defaulted customers are usually subject to individual provisioning, i.e. based on expert judgement on deal by deal basis."
- **Customer 2** - "Also a defaulted customer, i.e. PD is 100%. On the contrary, having zero RWF and RWA seems to be an issue. The transaction is not fully provisioned."
- **Customer 3** - "There seems to be an issue. I don't understand where OS_R is coming from, in [main dataset] I see zero outstanding. There are covers reported in [main dataset] in a similar amount (but not the exact amount). But also in [main dataset], I can see reported RWA_R (amount not matching figure in the table, but close). Having RWAs here seems to be an issue, I would not expect this for a zero limit with zero outstanding."

6.5 Explainable AI

Implementing Explainable AI through the Shapley Values detected the features that contributed the most to the detection of a specific record and the entire data sets for each of the Isolation Forest and Autoencoders models. In Table 5.2 it can be seen for the top anomaly record the higher contribution of RWA, PROVISIONS and PD features with a lower score for the Isolation Forest model. Outstanding (OS) is the feature with the highest contribution, for the third and fourth record, with a Shaply Value of -0.07, while Exposure at Default (EAD) shares the same value of -0.07 for the fourth row. Exposure is the feature with the second highest Shapley Value in the data set, with the highest contribution of -0.06 for the 8^{th} most anomalous record.

Figure 5.10 illustrates the overall contribution for the top 20 anomalies detected by Autoencoders. Here, the Loss Given Default (LGD), Risk Weight Factor (RWF) and Probability of Default (PD) are the features with the highest contribution in the output data set, calculated using a mean function.

Using, thus, Explainable AI the data analysts can easily spot and fix the data quality issues.

Chapter 7

Final Remarks

This chapter concludes the thesis paper, discussing its results, limitations, and possible directions for future work.

7.1 Conclusions

In this paper, we experimented with a parallel ensemble Machine Learning model, combining Isolation Forest and Autoencoders to detect anomalies and enhance the data quality detection of a credit risk data set inside a top banking institution (ING). By classifying anomaly detection algorithms from three perspectives, namely *methodology*, *outlier type*, and *technique*, this paper provides a comprehensive review of the current literature. The thesis then reviews and selects the most promising anomaly detection techniques discussed while performing experiments with two chosen methods. A Machine Learning model (Isolation Forest) and a Deep Learning model (Autoencoders) were used for experiments on an actual credit risk data set of one of Europe's most significant financial institutions.

While Isolation Forest exhibited robust performance, Autoencoders required several rounds of hyper-parameter tuning to achieve stability. Both models successfully detected potential data quality issues (anomalies) 100%, as confirmed by ING's internal data sets, which contain both random and systemic data quality issues.

Despite the promising results on the internal datasets anomaly detection success rate, the systemic data quality detection rate (2/6 true positives) remains suboptimal. These results are due to challenges in feature selection or insufficient hyperparameter tuning. Further adjustments to the model's parameters or incorporating domain-specific rules could help address this issue.

Considering the research goals identified in Chapter 1, this thesis implements a proof of concept that reduces the effort required by data analysts to detect data quality (DQ) issues. By leveraging a combined Machine Learning solution and Explainable AI, we provide a novel approach to improve both efficiency and interpretability in anomaly detection within banking institutions.

Answers to the Research Questions

RQ1: Many financial institutions still rely on manual, rule-based data quality detection, which is both time-consuming and prone to human error. Our findings demonstrate that AI-driven approaches significantly enhance efficiency by reducing execution time to under a minute, enabling real-time anomaly detection. Compared to traditional methods covered in [4],[6], [23] AI-based techniques also allow for greater scalability, automation, and adaptability in detecting diverse data quality issues.

RQ2: The automation of data management processes through AI improves the timeliness of data quality issue detection. This directly addresses the challenges outlined in McKinsey's

report [6], where periodic audits often introduce delays in identifying and resolving data issues. Our approach accelerates data quality management by continuously monitoring financial data, reducing dependency on scheduled audits, and enabling proactive corrections.

RQ3: One major limitation of traditional AI models in banking is their lack of interpretability [24]. To address this, we integrated Explainable AI techniques, providing insights into how anomalies are detected. This question is addressed through the integration of SHAP values and feature importance scores to provide transparency into model decisions. By coping with the black-box nature of Machine Learning models, analysts can better understand AI decisions and fine-tune detection thresholds, enhancing trust and usability in real-world financial applications.

RQ4: While the model demonstrates good anomaly detection performance, its computational complexity could be a concern for real-time deployment. A trade-off analysis indicates that the benefits in detection accuracy justify the increased resource demands, but optimizations for runtime efficiency are necessary. However, our research demonstrates the benefits of a parallel ensemble approach using Isolation Forest and Autoencoders, which enhances anomaly detection performance [42],[19]. The combined method allows for a comparative evaluation, leading to a more representative subset of anomalies being flagged for further validation.

RQ5: Existing classifications of anomaly detection methods are fragmented and inconsistent across different domains. To address this, we developed a structured taxonomy based on three perspectives: *methodology*, *outlier type*, and *technique*. By synthesizing insights from multiple sources [31, 34, 27], this thesis provides a standardized classification framework, improving the applicability of anomaly detection techniques across various financial and non-financial domains.

7.2 Limitations and Future Work

This thesis has the following limitations that could be addressed in future research:

1. **Limited number of algorithms** - The study contains experiments using only two Machine Learning methods: Isolation Forest and Autoencoders. These algorithms were selected based on their proven efficacy in anomaly detection tasks. However, the inclusion of additional algorithms could further improve the model's performance and robustness. In particular, a third selected algorithm, e.g. Generative Adversarial Networks (GANs), could offer significant advantages. GANs, with their complex structure and ability to generate synthetic data, are well-suited for anomaly detection in high-dimensional datasets, such as those typically found in banking. By leveraging the power of GANs to generate realistic data points, anomalies may be more easily identified. Furthermore, GANs can be combined with Autoencoders, as proposed by Du et al. [8], illustrated by Figure 3.7. This architecture could potentially enhance anomaly detection by improving the model's ability to differentiate between normal and anomalous data, particularly in cases of complex, nuanced data distributions.

2. **Root cause analysis** - In this current version, the models can accurately detect records that have potential data quality issues. However, in addition to detecting systemic data quality issues, the models must track the starting point of such issues to further improve the detection process in a banking institution due to the vast amount of data sets.

3. **Limited feedback from experts** - Although the method performed excellently in detecting potential (random) data quality issues from the commonly detected anomalies, only the first six records received feedback from the experts, making it hard to assess the method's efficiency. Moreover, due to time considerations, the final robust version of the model and the time-series data set (Dataset B') have not been managed to be validated by the experts. Thus, including a more significant amount of records in the data sets with expert feedback will improve the ability to assess the algorithms in detecting *systemic data quality issues* in particular.

4. **Limited training and test sets** - As the ensemble model has been trained and tested on a small chunk of the credit risk data set within ING, changing this approach to using more diverse data sets will improve the model's performance. This can be achieved by changing the training strategy to feeding the Autoencoder model with training and test sets from different data sets. This corresponds with the extraction of regular records in the autoencoders model.

5. **One ensemble learning method** - In this thesis, only the parallel ensemble learning method is implemented, where both models independently detect the anomalies. Implementing the ensemble method using the boosting or stacking method (see Section 3.2) could also generate better results by using a second model in the ensemble to train the errors made by the first model.

Finally, this study underscores the potential of Machine Learning methods for improving the detection of data quality issues in financial institutions. However, it also highlights the irreplaceable role of human expertise. Integrating this semi-automated system within a financial institution is a significant undertaking that requires time and resources, as safety is often prioritized over innovation. Therefore, implementing the suggested next steps, alongside conducting technical workshops within the bank to showcase the system's impact, is crucial to achieve the ultimate objectives.

Bibliography

[1] Marcel Altendeitering. Design Principles for Data Quality Tools. *Research Gate*, 2022.

[2] Van Gestel Tony & Baesens Bart. Credit Risk Management: Basic concepts: financial risk components, rating analysis, models, economic and regulatory capital. *ideas.repec.org*, 2008. URL: https://ideas.repec.org/b/oxp/obooks/9780199545117.html.

[3] Deepanshu Bhalla. A complete guide to credit risk modelling. URL: https://www.listendata.com/2019/08/credit-risk-modelling.html?form=MG0AV3.

[4] Antoon Bronselaer. Data quality management: an overview of methods and challenges, 2021. URL: https://biblio.ugent.be/publication/8721614.

[5] Victoria Collin. Moody's - Definition, How it Works, Credit Ratings Scale, 6 2024. URL: https://www.fe.training/free-resources/credit/moodys/.

[6] McKinsey & Company. Optimizing data controls in banking, 2020. URL: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/Risk/Our%20Insights/Optimizing%20data%20controls%20in%20banking/Optimizing-data-controls-in-banking-vF.pdf.

[7] Julianna Delua. Supervised vs. Unsupervised Learning: What's the Difference? - IBM Blog, March 2021. URL: https://www.ibm.com/blog/supervised-vs-unsupervised-learning/.

[8] Xusheng Du, Jiaying Chen, Jiong Yu, Shu Li, and Qiyin Tan. Generative adversarial nets for unsupervised outlier detection. Technical report, School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China and School of Software, Xinjiang University, Urumqi 830046, China, 2024. URL: https://doi.org/10.1016/j.eswa.2023.121161.

[9] Bernd Engelmann and Ho Chi Minh City Open University. A Simple and Consistent Credit Risk Model for Basel II/III, IFRS 9 and Stress Testing when Loan Data History is Short. Technical report, Ho Chi Minh City Open University, 3 2023. URL: https://ssrn.com/abstract=3926176.

[10] Ibomoiye Domor Mienye et al. A survey of ensemble learning: Concepts, algorithms, applications, and prospects, 2022. URL: https://ieeexplore.ieee.org/document/9893798.

[11] family=Farid given i=J, given=Jawwad. Unexpected loss (ul), expected loss, economic capital. URL: https://financetrainingcourse.com/education/2014/12/unexpected-loss-ul-expected-loss-economic-capital-case-study/.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, †, Aaron Courville, Yoshua Bengio, ‡, D´epartement d'informatique et de recherche op´erationnelle, and Universit´e de Montr´eal. Generative adversarial Nets.

arXiv:1406.2661v1 [stat.ML] 10 Jun 2014, 6 2014. URL: https://arxiv.org/pdf/1406.2661v1.pdf.

[13] IBM. What is data quality? | IBM. URL: https://www.ibm.com/topics/data-quality.

[14] IBM. What is ensemble learning? | IBM. URL: https://www.ibm.com/topics/ensemble-learning.

[15] Vladimir Bok Jakub Langr. GANs in action. URL: https://books.google.nl/books/about/GANs_in_Action.html?id=HojvugEACAAJ&redir_esc=y.

[16] Pranav Vigneshwar Kumar, Ankush Chandrashekar, and K. Chandrasekaran. *Machine Learning based Data quality Model for COVID-19 related big data*. Springer, 11 2021. doi:10.1007/978-981-16-6285-0_44.

[17] Gautam Kunapuli. *Ensemble Methods for Machine learning*. Simon and Schuster, 5 2023.

[18] F. Laakom et al. Reducing redundancy in the bottleneck representation of autoencoders. *SSRN*, 2022. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4255414 (Accessed: 04 January 2025).

[19] Fei Tony Liu, K. Ting, and Zhi-Hua Zhou. Isolation-Based anomaly detection, 2012. URL: https://www.semanticscholar.org/paper/Isolation-Based-Anomaly-Detection-Liu-Ting/dcd9f5d61bc9a70c40be84a8d78fbee822ffcd9e.

[20] COR-PAUL BEZEMER M. Sabuhi et. MING ZHOU and PETR MUSILEK. Applications of GANs in Anomaly Detection. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9631286.

[21] Batta Mahesh. Machine Learning Algorithms -A Review. Technical report, Smt. Kashibai Navale College of Engineering, Pune, India, 1 2020. URL: https://www.ijsr.net, doi:10.21275/ART20203995.

[22] Umberto Michelucci. An Introduction to Autoencoders. *TOELT.AI*, 1 2022. URL: https://arxiv.org/pdf/2201.03898.

[23] Helen-Tadesse Moges, K. Dejaeger, Wilfried Lemahieu, and Bart Baesens. Data quality for credit risk management: New insights and challenges proceedings. *ResearchGate*, 1 2011. URL: https://www.researchgate.net/publication/286811120_Data_quality_for_credit_risk_management_New_insights_and_challenges_proceedings#fullTextFileContent.

[24] Christoph Molnar et al. Interpretable machine learning. *arXiv preprint arXiv:1901.02891*, 2019. URL: https://arxiv.org/abs/1901.02891.

[25] Dimas Cassimiro Nascimento, Carlos Eduardo Santos Pires, and Demetrio Gomes Mestre. Applying machine learning techniques for scaling out data quality algorithms in cloud computing environments. *Applied intelligence*, 45(2):530–548, 4 2016. doi:10.1007/s10489-016-0774-2.

[26] Asif Ahmed Neloy and Maxime Turgeon. A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs. *Machine learning with applications*, page 100572, 7 2024. URL: https://www.sciencedirect.com/science/article/pii/S2666827024000483, doi:10.1016/j.mlwa.2024.100572.

[27] A. Patcha and J. M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007. doi:10.1016/j.comnet.2007.02.001.

[28] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A Design Science research Methodology for Information Systems research. *Journal of Management Information Systems*, 24(3):45–77, 12 2007. URL: https://doi.org/10.1007/978-981-16-6285/10.2753/mis0742-1222240302, doi:10.2753/mis0742-1222240302.

[29] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011. URL: https://www.researchgate.net/publication/228787694_Evaluation_From_Precision_Recall_and_F-Measure_to_ROC_Informedness_Markedness_and_Correlation.

[30] Abhinav Prakash. Different types of Autoencoders, 7 2019. URL: https://iq.opengenus.org/types-of-autoencoder/.

[31] Durgesh Samariya, B and Amit Thakkar. A comprehensive survey of anomaly detection algorithms. Technical report, Federation University, Churchill, VIC, Australia and Charotar University of Science and Technology (CHARUSAT), Gujarat, India, 11 2023. URL: https://doi.org/10.1007/s40745-021-00362-9.

[32] SciKit. IsolationForest. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[33] Mohit Sewak, Sanjay K. Sahay, and Hemant Rathore. An overview of deep learning architecture of deep neural networks and autoencoders. *Journal of computational and theoretical nanoscience*, 17(1):182–188, 1 2020. URL: https://www.ingentaconnect.com/contentone/asp/jctn/2020/00000017/00000001/art00029, doi:10.1166/jctn.2020.8648.

[34] K. Singh and S. Upadhyaya. Outlier detection: Applications and techniques. https://www.researchgate.net/publication/267964435_Outlier_Detection_Applications_And_Techniques, 2012.

[35] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. doi:10.1016/j.ipm.2009.03.002.

[36] Statology. Random errors vs. systematic errors: What's the difference?, 2021. Accessed: 2024-08-31. URL: https://www.statology.org/random-errors-vs-systematic-errors/.

[37] Constantinos Stephanou, Financial Stability Board / BIS, Juan Carlos Mendoza, and World Bank. Credit Risk Measurement under Basel II: An Overview and implementation issues for developing countries. *World Bank Policy Research Working Paper*, 4 2005. URL: https://www.researchgate.net/publication/23723087.

[38] Sudiksha, Preethi Nanjundan, and Jossy P. George. Machine learning classifiers for credit risk analysis. *EAI endorsed transactions on internet of things*, 10, 3 2024. URL: https://publications.eai.eu/index.php/IoT/article/view/5376, doi:10.4108/eetiot.5376.

[39] True Tamplin. Credit Risk | Definition, Types, Measurement, and Management, 11 2023. URL: https://www.financestrategists.com/wealth-management/investment-risk/credit-risk/.

[40] Xuhong Wang, Ying Du, Shijie Lin, Ping Cui, Yuntian Shen, and Yupu Yang. adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-based systems*, 190:105187, 2 2020. URL: https://arxiv.org/abs/1903.00904, doi:10.1016/j.knosys.2019.105187.

[41] Xiaodong Zhang, Yuan Yao, Congdong Lv, and Tao Wang. Anomaly credit data detection based on enhanced Isolation Forest. *The international journal of advanced manufacturing technology/International journal, advanced manufacturing technology*, 122(1):185–192, 4 2022. doi:10.1007/s00170-022-09251-8.

[42] Chong Zhou. Papers with Code - Anomaly Detection with Robust Deep Autoencoders, 8 2017. URL: https://paperswithcode.com/paper/anomaly-detection-with-robust-deep.

[43] Zhi-Hua Zhou. *Ensemble methods*. Chapman and Hall/CRC, 6 2012.