

AMERICAN UNIVERSITY OF BEIRUT

# Transactional Anomaly Detection using Machine Learning: Enhancing Financial Security and Operational Efficiency at IDS Fintech

by

Lara Baltaji

Advisor(s)

Dr. Sirine Taleb

A Capstone Project  
submitted in partial fulfillment of the requirements  
for the degree of Master's in Business Analytics  
to the Suliman S. Olayan School of Business  
at the American University of Beirut

Beirut, Lebanon  
August 2023

# An Abstract of the Capstone Project of

Lara Baltaji

for

Master's in Business Analytics (MSBA)

Title: Transactional Anomaly Detection using Machine Learning: Enhancing Financial Security and Operational Efficiency at IDS Fintech

This project tackles the crucial challenges linked to anomaly detection in financial data, where uncommon irregularities can result in substantial losses and diminished trust. The primary challenge in financial transactions lies in the increasing complexity of the data and the consequent rise in the likelihood of errors and faults. To mitigate these risks, a sophisticated anomaly detection model was developed in collaboration with Integrated Digital Systems Fintech (IDS Fintech), a pioneering provider of financial software solutions in the MENA market. The project's main focus was on developing an anomaly detection model that could effectively identify abnormal transactional behaviors within IDS Fintech's transactional financial data. To achieve this, data analytics and machine learning techniques were harnessed, with a particular emphasis on leveraging an unsupervised Auto-Encoder model. This model, operating without the need for labeled anomalies, effectively learned the underlying structure of normal data by compressing it into a lower-dimensional space and then reconstructing it. Through a comprehensive methodology involving data pre-processing, feature engineering, model testing, fine-tuning, and rigorous evaluation, the Auto-Encoder emerged as the optimal choice for the project's objectives. Its capacity to capture subtle deviations and patterns within the data made it well-suited for the domain, where anomalies manifest in diverse forms, from transaction timing to currency exchange rates to many more contexts. The results of the project showcase the Auto-Encoder's exceptional performance, successfully detecting anomalies while surpassing other unsupervised models. This achievement underscores the model's ability to provide IDS Fintech with an advanced monitoring system to safeguard its transactional processes and financial operations, enhancing operational efficiency and minimizing risks. However, a limitation of the chosen model is its lack of interpretability, as it only identifies the row containing the anomaly without specifying the anomaly's feature within the record. Future research work could explore explainable methods that offer insights into which specific feature within a record is anomalous, as well as techniques for suggesting corrected values.

# Table of Contents

1 Introduction .....	4
2 Background and Related Work.....	6
2.1 Anomaly Detection in Transactional Data .....	6
2.2 Types of Anomalies .....	6
2.3 Common Challenges in Anomaly Detection.....	7
2.4 Different Approaches for Anomaly Detection using Machine Learning .....	8
3 Methodology.....	10
3.1 Business Understanding .....	10
3.2 Data Understanding.....	12
3.3 Libraries Used .....	16
3.4 Data Preparation.....	17
3.4.1 Data Cleaning .....	17
3.4.2 Feature Engineering .....	18
3.4.3 Preprocessing Numerical Features.....	20
3.4.4 Feature Selection.....	20
3.5 Modelling and Fine-Tuning .....	22
3.5.1 Data Visualization using t-SNE Plot.....	22
3.5.2 Train-Test Split .....	23
3.5.3 Evaluation Metrics .....	23
3.5.4 Supervised Learning .....	24
3.5.5 Unsupervised Learning .....	26
3.6 Evaluation .....	29
3.7 Deployment .....	29
4 Results .....	30
4.1 Supervised Learning.....	30
4.2 Unsupervised Learning .....	37
4.3 Deployment on Streamlit .....	43
5 Discussion.....	44
5.1 Supervised Models .....	44
5.2 Unsupervised Models.....	44
5.3 Optimal Model for our Specific Objective.....	45
6 Conclusion.....	46
7 References .....	48

# 1 Introduction

In today's fast-paced financial landscape, accurate and error-free transactions are of utmost importance to ensure smooth and efficient financial operations. However, with the increasing complexity and volumes of financial transactions, the likelihood of errors and faults has also risen. These transactional anomalies can lead to significant financial losses, regulatory compliance issues, and eroded customer trust. Identifying and addressing these anomalies in a timely manner is critical for any financial institution.

This business analytics capstone project aims to leverage the power of data analytics and machine learning to detect and prevent transactional errors and faults in financial data. By developing an objective and robust anomaly detection model, financial institutions will be able to safeguard their transactional processes with a practical approach, mitigating potential financial risks, and enhancing overall operational efficiency.

Recognizing this critical need, IDS Fintech decided to collaborate with me on my business analytics capstone project in order to harness the potential of data analytics and machine learning for detecting and preventing anomalies in their transactional financial data. Integrated Digital Systems Fintech (IDS Fintech) is a leading provider of cutting-edge Financial Software Solutions in the MENA market. Specializing in portfolio management and trading solutions, IDS Fintech empowers financial companies, including investment banks, brokerage firms, funds and family offices, with state-of-the-art tools for effective management, automation, and monitoring of their investment processes.

At the heart of IDS Fintech's comprehensive suite of solutions lies Vestio, an advanced asset management system that revolutionizes the way financial institutions handle their investment operations. Vestio brings together a powerful combination of modelling, benchmarking, rebalancing, risk calculators, and ad-hoc compliance tools, providing a platform that assures seamless integration with trading engines, such as ROR, and depository containers, including accounting systems. However, a main problem at IDS Fintech is that throughout its asset management and transactional records, abnormal transactional behavior may be recorded which could usually lead to significant losses, regulatory compliance issues, and eroded customer trust if not detected and handled immediately.

Therefore, the primary goal of this collaborative project is to develop and implement a sophisticated anomaly detection model tailored to IDS Fintech's transactional financial data. By leveraging machine learning algorithms, this project aims to provide IDS Fintech's clients with an advanced monitoring system that identifies abnormal transactional behaviors and outliers effectively.

There exist a number of benefits for implementing an automated anomaly detection model. First of all, IDS Fintech can streamline its reconciliation and error detection processes, leading to increased operational efficiency. Second, early detection and mitigation of anomalies will minimize potential financial risks and safeguard the financial interests of IDS Fintech's esteemed clientele. Third, ensuring error-free and reliable transactions will instill greater confidence in IDS Fintech's customers, strengthening their trust in the company's comprehensive financial solutions. Lastly, it will contribute to regulatory compliance for IDS Fintech and its client, minimizing compliance-related risks.

This project will use a rich and comprehensive dataset collected from IDS Fintech's portfolio management and trading solutions database. This dataset encompasses a wide range of

transaction types, currencies, account types, and amounts, providing a strong foundation for the development of an accurate anomaly detection model.

To achieve the project's objectives, several approaches will be conducted, including data pre-processing, feature engineering, model testing, fine-tuning, rigorous evaluation and model selection. The dataset will be thoroughly analyzed to identify relevant features and patterns, ensuring that the selected machine learning algorithms can efficiently detect anomalies in real-world financial transactions. An array of supervised and unsupervised machine learning algorithms will be explored to identify the most suitable model for detecting anomalies effectively. These models include: Logistic Regression, K-Nearest Neighbors, Gaussian Naïve Bayes, Support Vector Machine, Decision Tree, Random Forest, Extreme Gradient Boosting, Artificial Neural Networks, Isolation Forest, Local Outlier Factor, One-Class SVM, and Auto-Encoder.

One significant challenge that this project will address is that the data is not initially labelled. As such, employing supervised learning methods for anomaly detection and evaluating the performance of unsupervised learning techniques may not be immediately feasible. To overcome this issue, expert knowledge and domain expertise will play a crucial role in generating synthesized anomalies for implementing supervised learning and evaluation purposes. In addition, combining expert input with unsupervised learning techniques will establish a reliable baseline for evaluating the performance of the anomaly detection models.

This ambitious business analytics capstone project aims to provide IDS Fintech with an unmatched value for its clients in the MENA market. By harnessing the power of machine learning and data analytics, IDS Fintech will be able to empower clients with superior anomaly detection capabilities, reinforcing its position as a leading provider of financial software solutions and driving the financial industry into a future of secure and efficient transactions through automation.

## 2 Background and Related Work

This section presents a comprehensive background and literature review on anomaly detection in the finance industry, exploring the various methods used, the challenges faced, and the significance of automation in this critical domain.

Section 2.1 outlines the definition of anomaly detection in existing literature, the different applications of anomaly detection in financial data. Section 2.2 introduces the taxonomy of anomalies and Section 2.3 presents the most common challenges faced during anomaly detection implementation in transactional data in the financial sector. Finally, Section 2.4 outlines the different approaches for anomaly detection in finance, each with its own challenges and limitations. It also presents approaches to synthetically generate anomalous data for a better performance.

### 2.1 Anomaly Detection in Transactional Data

*Anomaly detection* is described in literature as identifying irregularities that deviate from anticipated patterns. According to Chandola et al. (2009), these atypical patterns are frequently labeled as anomalies, outliers, unexpected observations, exceptions, aberrations, surprises, peculiarities or contaminants across various fields. Among these terms, anomalies and outliers are the most commonly used interchangeably (Chandola et. al., 2009). Anomaly detection methods typically revolve around establishing a clear definition for normal data and recognizing any deviations from this norm (Anandakrishnan et. al., 2017).

The increasing use of technology has reformed the modern financial industry through the use of data analytics and machine learning techniques to learn valuable insights from financial data for better decisions. The use of anomaly detection in the financial sector mainly serves for fraud detection, risk management and predictive analysis purposes (Nourbakhsh & Bang, 2019). For example, if the IP address of a client doing a transaction has changed compared to the last transaction, this may indicate an anomaly. It may imply a risk of fraud, a technical error, an entry error, or it may simply imply that the client has changed location. According to Anandakrishnan et al. (2017), applications for financial data anomaly detection include tasks such as detecting transactional fraud, fighting money laundering, addressing identity theft and fraudulent account registrations, developing risk models, mitigating promotion abuse, conducting customer behavior analytics and enhancing cybersecurity. No matter what specific purpose the anomaly detection has, the primary aim is to ensure the financial system's integrity and security (Liu et. al., 2023).

According to an article written by Ahmed et al. (2016), financial sectors generate huge amounts of data that follows the three main big data properties, which are volume, velocity and variety. The use of machine learning (ML) for detecting anomalies in financial data aims to reduce manual work of auditors, minimize the likelihood of errors and lower the risk of material misstatement risk reduction when sampling and evaluating a large volume of financial transactions (Bakumenko & Elragal, 2022). In contrast to rule-based anomaly detection, which relies on manually designed algorithms, ML approaches offer advantages through their capacity for automated learning and the identification of concealed patterns (Bakumenko & Elragal, 2022).

### 2.2 Types of Anomalies

One crucial factor in anomaly detection is the type of anomaly one seeks to detect. According to Chandola et al. (2009), anomalies can be categorized into three types:

- *Points anomalies* which are the simplest types of anomalies are usually the focus of most research. In this scenario, a single feature of an observation is affected. Thus, we should treat the feature independently for the rest. For example, in a transactional data, a transaction where the amount paid is much higher than the normal range of expenditure is considered as a point anomaly.
- *Contextual anomalies* expand upon point anomalies by considering multiple features simultaneously, assuming interdependencies between them. A data instance could be considered a contextual anomaly in a certain context but normal in a different context. Contextual anomalies treat each observation independently. Continuing with the previous example, a contextual attribute could be the location of the applicant. A purchasing amount of \$1500 might be deemed too high in one country but considered as a normal behavior in another based on the average salary differences.
- *Collective anomalies* are identified through the examination of groups of observations. Contextual anomalies treat each observation independently, while collective anomalies emerge when analyzing a few observations together. Each individual observation might not appear anomalous, but a cluster of such observations occurring in cycles defines the anomaly. This often arises when considering the time-based aspect. For instance, consider a scenario where rules prohibit spending beyond one's credit limit. In this case, if an individual exceeds their credit limit within a week, this behavior may seem normal in isolation, treating each observation separately. However, when observed collectively, it transforms into an abnormal activity, thus constituting a collective anomaly.

The concept of a collective anomaly falls beyond the scope of this study, as it necessitates significantly different approaches compared to those used for the first two anomaly types.

### 2.3 Common Challenges in Anomaly Detection

Several factors make the detection of anomalies in financial data very challenging, which include:

- *Difficulty in defining normality*: Establishing a clear definition of the normal range and the anomalous range for all potential behaviors presents a substantial challenge (Chandola et. al., 2009). As the number of features in a dataset increase, the challenge becomes more difficult due to the curse of dimensionality, as mentioned by Daniel Schlör (2022).
- *Lack of labeled data*: According to Anandakrishnan et al. (2017), it is usually very difficult to find or generate labeled data that includes all anomalies. Humans can manually identify a limited number of anomalies. However, this process is time-consuming, especially that the data is huge, and requires expert knowledge (Morozov, 2016). Even if labels are available for some anomaly categories, dynamic occurrences like fraud and money laundering are constantly evolving, and numerous situations arise where creating anomalies becomes impossible (Anandakrishnan et. al., 2017). This presents another issue related to the training and evaluation models. Unsupervised machine learning techniques offer a primary solution to address this challenge. Another possible solution is to add simulations of anomalous entries based on expert knowledge. An extensive exploration of these two approaches will be presented in the subsequent section.
- *Class Imbalance*: Scarcity of anomalies ultimately results in class imbalance, making machine learning process difficult. Some methods to deal with this challenge include using metrics other than accuracy for evaluation or considering oversampling techniques (Crépey et. al., 2022). These two approaches will also be presented in the following section.

Before delving into the specific statistical, machine learning (ML), and deep learning (DL) techniques extensively employed for anomaly detection within the financial sector, let us initially examine the various methods used to overcome the challenges outlined in the preceding section.

To deal with the challenge of unlabeled data, researchers have explored different solutions. Bakumenko and Elragal (2022) decided to assume normality of the original data, while adding a small amount of synthesized anomalous entries provided by financial experts. While the original data might encompass real anomalies, their count is presumably very low, thus having a negligible impact on the evaluation process. The synthesized anomalies, on the other hand, are seen by experts as anomalies of interest because they represent particularly rare occurrences with the potential to result in significant losses. Similar methods of adding simulations of anomalous entries have been investigated by other researchers such as Daniel Schöler (2022), Becirovic et al. (2020) and Crépey et al. (2022). Other scholars refrained from labeling the data and instead focused on employing unsupervised learning techniques, which do not require labeled data, for identifying abnormalities. For example, Li and Jung (2021) decided to consider specific data points as outliers through distance-based and statistical algorithms as a means of evaluating their unsupervised models.

The issue of imbalanced classes could be addressed in many ways. One well-known method is to use oversampling methods such as SMOTE and ADASYN (Bakumenko & Elragal, 2022). Yıldız et al. (2022) and Daniel Schlör (2022) both used the Synthetic Minority Oversampling Technique (SMOTE) to produce an equal class distribution of their data, hence reducing the effect of class imbalance on the model training. Others decided to use class-sensitive learning by passing class weights to models to compensate for class imbalance, such as Ivan Morozov (2016) and Bakumenko and Elragal (2022). The final and most common method for addressing the imbalance of classes is to simply consider evaluation metrics other than accuracy such as the ROC-AUC, Precision, Recall, F1-score and False Positive Rate (FPR).

## **2.4 Different Approaches for Anomaly Detection using Machine Learning**

The use of statistics, machine learning (ML) and deep learning (DL) algorithms for the aim of detecting anomalies in financial data has been extensively studied, especially during the past few years. Several methods are employed for anomaly detection in finance, each with its own challenges and limitations.

Prior to the advent of machine learning, statistical approaches were extensively employed for detecting outliers. Statistical methods involve fitting a statistical model, typically representing typical behavior, to provided data. Subsequently, a statistical inference test is employed to check whether an unfamiliar instance aligns with this model. Instances with a minimal likelihood of being generated by the established model, as indicated by the applied test statistic, are identified as anomalies (Chandola, 2009). Parametric statistical techniques assume that data provided follows a stable parametric distribution and anomalies are data points that lie outside this distribution (Chandola, 2009). Some of the most commonly used parametric techniques include z-score test, box-plot test, basic linear regression model and Autoregressive Integrated Moving Average (ARIMA) model. Non-parametric statistical techniques, on the other hand, do not make any assumptions related to the data structure, where the statistical model generates a structure from the data itself (Chandola, 2009). These include histogram-based and kernel-function based. Histogram-based techniques create univariate histograms with bins of frequencies for every feature. An anomaly would be a test value falling outside any one of the bins. Kernel-function based techniques work on estimating the actual density of the data using kernel functions. Any value lying in the low probability area would be anomalous (Chandola, 2009).



Machine learning techniques are divided into three parts, supervised, unsupervised and semi-supervised. In supervised learning, the presence of labeled data instances is required, enabling the model to learn the intended output (Bakumenko & Elragal, 2022). Unsupervised learning algorithms, however, do not require labeled data to be trained. Such techniques aim to learn the structure of the data without generating classes or numerical estimates. These include distance and density-based methods for clustering data points which have similar properties. Labels can still be used for evaluating the performance of unsupervised models (Bakumenko & Elragal, 2022). The choice of a machine learning model for anomaly detection relies on factors such as label availability and specific needs for uncovering abnormal patterns.

Anomaly detection requires classifying data entries as normal or anomalous, making it a classification model. Supervised machine learning algorithms aim to learn the patterns of the different normal and anomalous data and thus predicting labels, which necessitates the existence of labels. Some commonly employed machine learning classifiers for anomaly detection in the financial sector include Logistic Regression (LR), Support Vector Machines (SVM), Decision Tree-oriented classifiers, K-Nearest Neighbor (k-NN), Naïve Bayes, Gradient Boosting classifier such as XG-Boost, and Deep Neural Networks (Bakumenko & Elragal, 2022).

For example, Daniel Schöler (2022) tested eight supervised ML models on 4 datasets. The best-performing models turned out to be SVM, Logistic Regression, SVM and XG-Boost, with very close results. Bakumenko and Elragal (2022) also trained seven supervised model to detect anomalies in general ledger (GL) financial data, which are Logistic Regression, SVM, Decision Tree, Random Forest, KNN, Naïve Bayes and Deep Neutral Networks. They trained three neural networks which are the 'Vanilla' ANN with one hidden layer, DNN with one hidden layer and another DNN with two hidden layers. Out of all three NN models, the DNN with three hidden layers gave the highest macro-averaged recall of 0.9277. Out of all supervised model. the Random Forest model resulted in the highest macro-averaged recall score of 0.9925.

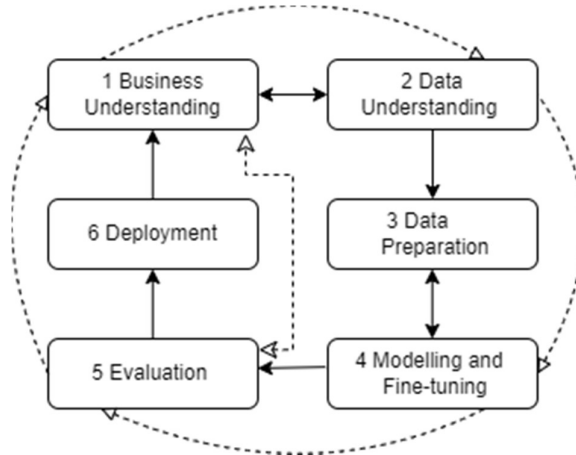
Conversely, unsupervised learning techniques require no label during the model fitting procedure. Every data instance is considered as normal or anomalous based on the training features alone. Such models use similarity to cluster objects into one big group and every object lying outside the cluster is considered anomalous (Omar et. al., 2013). They serve as an advantage because they can be applied to unlabeled data, thus forcing less requirements and assumptions, while still being evaluated on a limited number of well-known anomalous entries using a the True Positive (TP) score, for example. The mostly frequently used unsupervised learning techniques for detecting anomalies in financial data are K-means Clustering, One-class Support Vector Machine (OC-SVM), Isolation Forest (IF), Local Outlier Factor (LOF), Density-based Spacial Clustering of Applications with Noise (DBSCAN) and Auto-encoders.

In the same article mentioned earlier, Daniel Schöler (2022) also tested four unsupervised models. First, he tested two types of Auto-encoders, one using mixed layers proposed by the author, which uses 50% non-linear hidden units (ReLU) and 50% improve Neural Arithmetic Logic Unit (iNALUs), and the second using only ReLU Layers. He, then, tested the IF and the OC-SVM. The average results of the unsupervised training on the three used datasets show that the IF and the Mixed Layers performed the best among all four, with IF earning a ROC-AUC of 0.89 and the Mixed Layers earning an F1-score of 0.84. Additionally, Bakumenko and Elragal (2022) also experimented two unsupervised learning models, IF and Auto-encoders. The results of their study reveal that the IF was able to capture more TP values, with a macro-averaged recall score of 0.9555, in comparison to 0.9441 for the Autoencoders.

# 3 Methodology

In this section, we describe the methodology used to develop our transactional anomaly detector. Our process followed the CRISP-DM methodology cycle, as illustrated in **Figure 1**. The CRISP-DM framework, which stands for Cross-Industry Standard Process for Data Mining, comprises six distinct stages that start with understanding the business requirements and conclude with deploying the finalized solution. The tasks related to data are managed during the phases of understanding the data, preparing it, and constructing models, all of which are then evaluated (Bakumenko & Elragal, 2022).

The progression through these steps is not necessarily linear; iterations can occur from the initial business understanding to the evaluation stage. **Figure 1** shows the crucial phases involved in the development of a data science project. Particularly, there are two-way arrows indicating that adjustments can flow from a preceding stage to a subsequent one and vice versa. We will now explain the methodology based on the framework depicted in this diagram.

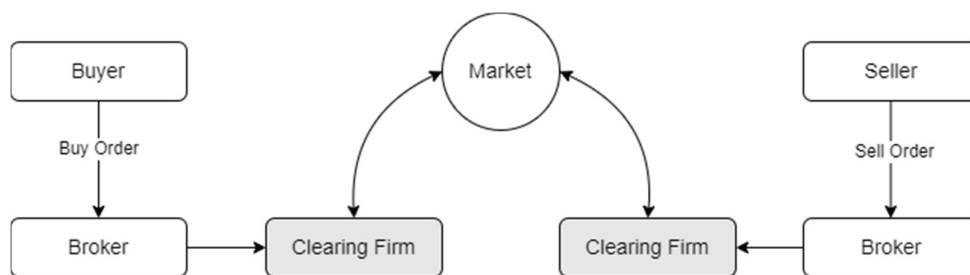


**Figure 1.** CRISP-DM Methodology Cycle

## 3.1 Business Understanding

As seen in **Figure 2**, the stock trading process involves multiple parties, each playing a crucial role in facilitating the exchange of stocks between buyers and sellers.

- *The Client*: The client, often an individual or institutional investor, initiates the process by expressing their interest in buying or selling a stock.
- *The Broker*: The broker acts as an intermediary between the client and the market. The client communicates their trading intention to the broker, who then executes the trade on their behalf. Brokers can provide valuable insights, research, and advice to clients, helping them make informed decisions.
- *The Clearing Firm*: The clearing firm ensures the smooth exchange of securities and funds between the buyer and seller. It handles the settlement process, verifying ownership and arranging for payment.
- *The Market*: The market can refer to various stock exchanges, such as the New York Stock Exchange (NYSE) or Nasdaq.



**Figure 2: Trading Process**

The trading process goes as follows:

1. *Client (Buyer/Seller) Decision:* A client, also known as an investor, decides to buy or sell a particular stock. This decision is based on various factors such as market analysis, company performance, and personal financial goals.
2. *Placing an Order:* The client contacts their chosen brokerage firm to place an order. The order specifies details such as the stock's symbol (ticker), the desired quantity, and whether the order is to buy (bid) or sell (ask) the stock. The client can place different types of orders, such as market orders (executed immediately at the current market price) or limit orders (executed only at a specific price or better).
3. *Brokerage Firm:* The brokerage firm acts as an intermediary between the client and the market. They receive and process the client's order. The order is then sent to the market for execution through various trading platforms and exchanges.
4. *Order Routing:* The brokerage firm employs sophisticated technology to route the client's order to the most appropriate exchange or trading venue. This is done to ensure the best execution for the client, seeking the best available price and liquidity.
5. *Market Execution:* Once the order reaches the market, it interacts with other orders from different market participants, including other investors, institutional traders, and market makers. The stock's price is determined by the interactions between supply and demand for that particular stock.
6. *Matching Buyers and Sellers:* The market's order matching engine matches buy and sell orders based on their price and time priority. This ensures that trades are executed fairly and efficiently.
7. *Trade Confirmation:* Once the trade is executed, the brokerage firm sends a trade confirmation to the client, detailing the transaction's price, quantity, and other relevant information.
8. *Clearing Firm:* After the trade is executed, the clearing process begins. The clearing firm steps in to facilitate the settlement of the trade. They ensure that both the buyer and the seller fulfil their obligations in the transaction.
9. *Trade Settlement:* The clearing firm handles the movement of securities (stocks) and funds between the buyer's and seller's accounts. Settlement involves the transfer of ownership of the stocks from the seller's account to the buyer's account and the transfer of funds from the buyer to the seller.
10. *Account Updates:* Once the settlement is complete, the brokerage firm updates the client's account to reflect the executed trade. The client's portfolio balance and overall account value are adjusted accordingly.

It is important to note that the entire process from placing an order to trade settlement happens within a relatively short timeframe, often in a matter of seconds or minutes, due to the high-speed electronic nature of modern financial markets. The involvement of brokerage firms, clearing firms, and technology infrastructure helps ensure the smooth and efficient functioning of the stock trading process. Throughout this trading process, each party involved charges fees for their services. Broker

fees compensate brokers for their services and expertise, exchange fees cover the cost of using the trading platform, and settlement fees support the clearing firm's role in facilitating the exchange. These fees can vary and impact the overall cost and profitability of the trade.

In summary, the stock trading process involves the client initiating a trade through a broker, which is then passed on to a clearing firm for settlement. The trade ultimately takes place on the market, where stocks are exchanged between buyers and sellers. This collaborative effort between multiple parties ensures the efficient and secure trading of stocks while incurring associated fees for their services.

At Integrated Digital Systems Fintech (IDS Fintech), such trading processes and financial portfolios are organized and managed. IDS Fintech is a leading company in the MENA region, offering advanced financial software solutions. They focus on portfolio management and trading solutions, equipping financial institutions like investment banks, brokerages, funds, and family offices with advanced tools to efficiently manage, automate, and track their investment activities. IDS Fintech charges its clients with a minimal charge as exchange for its services. This charge is known as the *Ticket Charge*, which is a feature present in the dataset provided by IDS Fintech.

### 3.2 Data Understanding

The dataset collected from IDS Fintech describes various cash financial trading transactions involving the buying and selling of different assets. Each row in the data represents a single transaction with different attributes (fields) associated with it. **Table 1** the fields of the dataset with their different data types. The transactional tickets span over six months, from January 1, 2023 till May, 31, 2023.

1. *TypeId*: Type of the transaction (11 for purchase, 12 for sale).
2. *SubTypeId*: Subtype of the transaction (further classification within the Type).
3. *MethodId*: Method used for the transaction (e.g., cash, bank transfer, etc.).
4. *HoldingId*: Holding identifier, related to the stock/asset being traded.
5. *ClientId*: Identifier for the client associated with the transaction.
6. *ProductId*: Identifier for the financial product involved in the transaction.
7. *BrokerId*: Identifier for the broker platform involved in the transaction.
8. *TradingAccountId*: Identifier for the trading account used in the transaction.
9. *ShareAccountId*: Identifier for the share account used in the transaction.
10. *CashBankAccountId*: Identifier for the cash bank account used in the transaction.
11. *CashThirdPartyAccountId*: Identifier for the third-party cash account used in the transaction if applicable.
12. *Quantity*: The quantity of the financial product involved in the transaction.
13. *UnitPrice*: The unit price of the financial product.
14. *AVGCost*: The average cost of the financial product.
15. *ExchangeFees*: Fees associated with the exchange for the transaction.
16. *BrokerageFees*: Fees charged by the broker for the transaction.
17. *TicketCharges*: Fees charged by the company for placing the ticket.
18. *SettlementFees*: Fees related to the settlement of the transaction.
19. *OtherFees*: Any other fees associated with the transaction.
20. *ValueCurrencyId*: Identifier for the currency used for the transaction value.
21. *ValueAmount*: The amount of the transaction in the value currency.
22. *CashSettlementDate*: The date of cash settlement for the transaction.
23. *ShareSettlementDate*: The date of share settlement for the transaction.

24. *LocalValueRate*: The local value rate.
25. *LocalEquivalent*: The local equivalent amount.
26. *CreationDate*: Date of the transaction creation.
27. *Anomaly*: Binary field showing whether the specific entry is presumed to be anomalous (1) or not (0).
28. *Anomaly Type*: The specific type of the anomalous entry.

**Table 1:** Dataset Features with Data Types

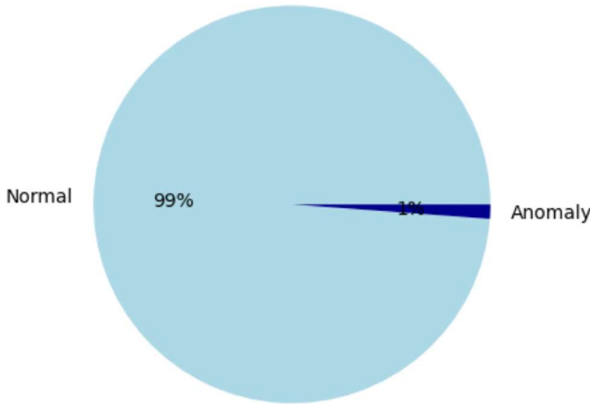
<i>Field</i>	<i>Data Type</i>
TypeID	int
SubTypeId	int
MethodId	int
HoldingId	int
ClientId	int
ProductId	int
BrokerId	int
TradingAccountId	int
ShareAccountId	int
CashBankAccountId	int
CashThirdPartyAccountId	int
Quantity	float
UnitPrice	float
AVGCoS	float
ExchangeFees	float
BrokerageFees	float
TicketCharges	float
SettlementFees	float
OtherFees	float
ValueCurrencyId	int
ValueAmount	float
CashSettlementDate	datetime
ShareSettlementDate	datetime
LocalValueRate	float
LocalEquivalent	float
CreationDate	datetime
Anomaly	int
Anomaly Type	int

The original dataset collected from IDS Fintech is assumed to be normal and lacking anomalous entries, while it may contain roughly up to 1-2 percent anomalies. In addition to this dataset, a few synthetically-generated anomalous entries were introduced into the original dataset in order to provide an improved evaluation of the models' performance. There are various types of anomalies, where each anomalous entry falls under a specific category. These specific anomaly types have been chosen based on the priority of the company's auditors, as they have been periodically seen but not directly recorded. Anomalies present in the original data may encompass not only the seven predefined anomaly types, but also other deviations from normal patterns. The

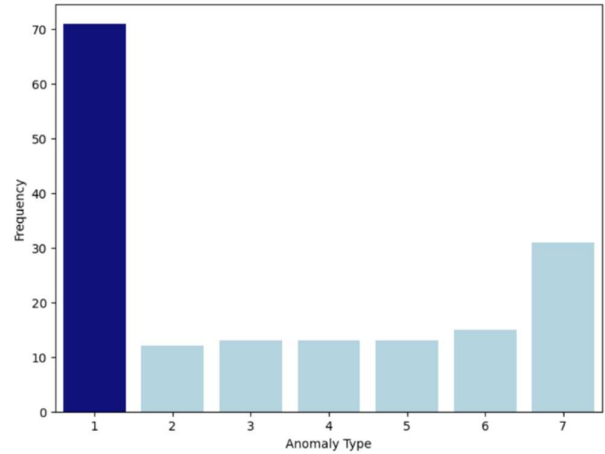
*AnomalyType* field may contain values from 0 to 7, where a normal entry will have a type 0 and the remaining correspond to one of the types defined by the company’s auditors. **Table 2** shows the counts of original dataset and the synthesized anomaly types, ordered in descending order of the count, **Figure 3** shows a pie chart of normal and anomaly percentages and **Figure 4** shows bar plot of the anomaly label type, excluding type 0 which are normal entries.

**Table 2:** Normal and Anomaly Type Counts

<i>Anomaly Type</i>	<i>Count</i>
Normal Data (Anomaly Type 0)	14301
Anomaly Type 1	71
Anomaly Type 7	32
Anomaly Type 6	15
Anomaly Type 3	14
Anomaly Type 5	14
Anomaly Type 4	13
Anomaly Type 2	12



**Figure 3:** Pie Chart of the Normal vs Anomaly Label



**Figure 4:** Bar Plot of the Anomaly Type Counts

There are 14471 transactions in total, where 14301 of them are collected from the original dataset, and are assumed to be normal. There are 170 synthetically generated anomalous entries, each with a count between 71 and 12. **Table 3** shows the counts of normal and anomalous data.

**Table 3:** Normal and Anomalous Counts

<i>Label</i>	<i>Count</i>
Normal Data (Label 0)	14301
Anomaly (Label 1)	170

The dataset consists of a total of 14,301 normal entries and 170 anomalous entries, with the anomalies accounting for approximately 1.2% of the entire dataset. This distribution highlights a

significant imbalance in the dataset as seen the pie chart in *Figure 3*. The descriptions of the different anomaly types are presented in *Table 4*.

**Table 4:** Description of synthesized anomalies

<i>Anomaly Type</i>	<i>Description</i>
1	Transaction occurring outside market working days and working hours
2	Cash settlement date occurring before share settlement date
3	Unit price of a product deviating significantly from its average cost
4	Ticket charge deviating significantly from the usual ticket charge pattern
5	Local settlement rate deviating significantly from the usual local exchange rate pattern for the same currency, within a small-time frame
6	Product quantity purchase/sale deviating significantly from the usual product quantity purchase/sale pattern
7	Negative quantity or value amount which is a classic mistake

As a reminder of the types of anomalies, point anomalies are identified purely based on the characteristics of individual data points, while contextual anomalies require the consideration of additional contextual information to determine their anomaly status. Based on these definitions, let us categorize each of our seven anomaly types as point or contextual anomalies.

- Anomaly Type 1: Contextual Anomaly

This anomaly's status depends on the contextual information of market working days and hours. A transaction occurring outside these expected timeframes would be considered anomalous within that context.

- Anomaly Type 2: Contextual Anomaly

This anomaly's status is determined by the relationship between cash settlement and share settlement dates. Depending on the context, a cash settlement date preceding a share settlement date could be considered anomalous.

- Anomaly Type 3: Point Anomaly

This anomaly is identified based on the individual data point's deviation from the average cost of the product. It doesn't require contextual information beyond the unit price and average cost.

- Anomaly Type 4: Point Anomaly

Similar to the previous case, this anomaly is identified based on the individual ticket charge's deviation from the usual pattern, without requiring additional contextual information.

- Anomaly Type 5: Contextual Anomaly

This anomaly's determination involves the context of exchange rates. A local settlement rate deviating significantly from the usual exchange rate pattern for the same currency is considered anomalous.

- Anomaly Type 6: Point Anomaly

Like the previous point anomalies, this anomaly is identified by the individual data point's deviation from the usual pattern of product quantity purchase or sale.

- Anomaly Type 7: Point Anomaly

This anomaly is identified solely based on the characteristics of the data point (negative quantity or value amount). It doesn't involve contextual factors beyond the specific data point itself.

### 3.3 Libraries Used

All the methodology steps were made using *Python* programming language. The preprocessing and modelling part were written in a *Jupyter Notebook*, while the deployment part was done using *Spyder* IDE. The analysis and modeling process conducted in this study make use of various libraries in *Python* to facilitate data manipulation, preprocessing, visualization, modeling, evaluation, and more. **Table 5** outlines the key libraries employed in this methodology along with their specific usages.

**Table 5:** Different Libraries Used

<b>Library</b>	<b>Usage and Purpose</b>
<i>Pandas</i>	Data manipulation and analysis, including data loading, transformation, and exploration.
<i>Numpy</i>	Fundamental package for numerical computations, used for mathematical operations and array manipulation.
<i>Matplotlib</i>	Data visualization, used to create static, interactive, and publication-quality visualizations.
<i>Seaborn</i>	Statistical data visualization based on matplotlib, provides aesthetically pleasing visualizations.
<i>Scikit-learn</i>	Scikit-learn library for machine learning, used for pre-processing, modelling, and evaluation.
<i>TensorFlow</i>	Open-source deep learning framework, employed for building and training neural network models. Used for
<i>PyOd</i>	Python Outlier Detection (PyOD) library for outlier detection algorithms and evaluation.
<i>Xgboost</i>	Extreme Gradient Boosting library, utilized for building gradient boosting machine learning models.
<i>KerasTuner</i>	Hyper parameter tuning library for optimizing neural network architectures using different algorithms.
<i>Pickle</i>	Package used for saving pipelines, transformers and models to be used for deployment on web frameworks like Streamlit.
<i>Streamlit</i>	Streamlit is a free and open-source Python library that makes it easy to create beautiful, custom web apps for machine learning and data science.

Data Preprocessing:

- *StandardScaler* and *MinMaxScaler* from *sklearn.preprocessing*: Used for scaling features to a standard range and minimum-maximum range, respectively.

Modelling:

- Various classifiers such as *RandomForestClassifier*, *DecisionTreeClassifier*, *LogisticRegression*, *SGDClassifier*, *KNeighborsClassifier*, *GaussianNB*, *SVC*, *DBSCAN*, *XGBClassifier* from *sklearn* and *xgboost* libraries for building different types of supervised machine learning models.
- *tf.keras* from *tensorflow* for creating and training supervised neural network models and unsupervised auto-encoder model.



- Unsupervised machine learning algorithms including *LOF*, *OCSVM*, and *IForest* from *pyod* for unsupervised anomaly detection in the data.

Evaluation:

- Metrics such as *confusion\_matrix*, *classification\_report*, *accuracy\_score*, *roc\_curve*, *roc\_auc\_score*, *recall\_score*, *mean\_squared\_error* from *sklearn* and *pyod* to evaluate model performance.

Hyperparameter Tuning:

- *GridSearchCV* from *sklearn.model\_selection* for exhaustive grid search over a specified parameter grid.
- Hyperparameter tuning using *kerastuner* for optimizing neural network architectures.

Data Visualization and Exploration:

- *scatter\_matrix* from *pandas.plotting* to visualize t-SNE plot in the dataset.
- *matplotlib* and *seaborn* for creating various types of plots and charts to explore and visualize the data.

Dimensionality Reduction and Feature Selection:

- *TSNE* from *sklearn.manifold* for t-Distributed Stochastic Neighbor Embedding, a technique for high-dimensional data visualization.

Model Deployment:

- *pickle* for saving and uploading trained models, transformers and pipelines to be used in the deployment of the anomaly detector on Streamlit.
- *Streamlit* for creating a user-friendly web application for testing the selected anomaly detector.

### 3.4 Data Preparation

All the below preparation steps were fitted using a pipeline to the training dataset, then transformed to the training and validation datasets in order to avoid data leakage.

#### 3.4.1 Data Cleaning

We started our preparation by checking the data for missing values and duplicated records. Although there were no null values, we did encounter 35 duplicated rows. We dropped the duplicated to improve the efficiency of our models and ensure that they are not fed any redundant information which could potentially impact their performance.

We also performed a review of the data types associated with each feature, which all turned out to be aligned with their original formats. Specifically, discrete numerical and binary values were stored as *int64* types, continuous variables were represented as *float64* types and date and time features became *datetime64* types.

### 3.4.2 Feature Engineering

Feature engineering involves extracting new features and transforming existing ones in order to improve the performance of machine learning models. In this subsection, we will focus on the feature extraction part. The different features that were extracted were primarily based on the different anomaly types.

Anomaly type 1 includes all transactions that occurred outside working days (all days except Friday which is the only official weekend in Kuwait) and working hours (8AM – 6PM). So, in order to detect such anomalous transactions, we used the *CreationDate* to extract the hour and time of transaction order. Then, we created a binary field being 1 if a transaction is occurring outside of working time and 0 otherwise using an “if-else” statement function.

Anomaly type 2 includes all transactions where cash settlement date occurred before share settlement date. In a typical market stock trade, stocks are settled before the settlement of cash. To detect those suspicious transactions, a new field was created which is simply the difference between the *ShareSettlementDate* and the *CashSettlementDate* in days. Then, another field being 1 if the difference is strictly negative (suspected anomaly) and 0 if positive (normal). The new extracted fields are *hour*, *weekday*, and *outside\_working\_time*.

For anomaly type 3, we wanted to compare between two fields which are the *UnitPrice* and the *AVGCost* field. If a transaction included a product unit price which deviates significantly from the typical average cost of this same product, then it is considered anomalous. So, a new feature was created which calculated the ratio between the *UnitPrice* and the *AVGCost*, and added a constant value to the denominator to ensure that the ratio remains well-defined even in cases where the *AVGCost* approaches zero, which would otherwise result in division by zero and yield values close to infinity. Now, if the ratio is close to 1, then the transaction would be considered normal. However, if the ratio is really big, then it would be considered as an anomaly. The new extracted fields are *date\_difference* and *negative\_date\_difference*.

Anomalies of type 5 are transactions whose local settlement rate is deviating significantly from the usual local settlement rate for the same currency. For example, if the typical currency exchange rate from USD to KWD (Kuwaiti Dinar) is between 0.3 and 0.31, then any transaction in the USD currency using a significantly different exchange rate will be suspicious. Therefore, by grouping transactions based on the *ValueCurrencyId*, then finding outliers in the *LocalSettlementRate* field, we can detect the suspicious anomalies, and that is what we did. The new extracted field is named *localvaluerate\_outliers*.

Finally, anomaly type 7 are anomalies where the *Quantity* or the *ValueAmount* are negative, which are classic mistakes in financial data. Since the data does not include a column showing if the transaction is reversed, then we should consider that all the trades have been successfully exchanged and any negative quantities or value amounts are to be considered doubtful. Therefore, a new binary feature, named *negative\_quantity\_valueamount*, was extracted showing 1 for negative quantities and value amounts and 0 otherwise.

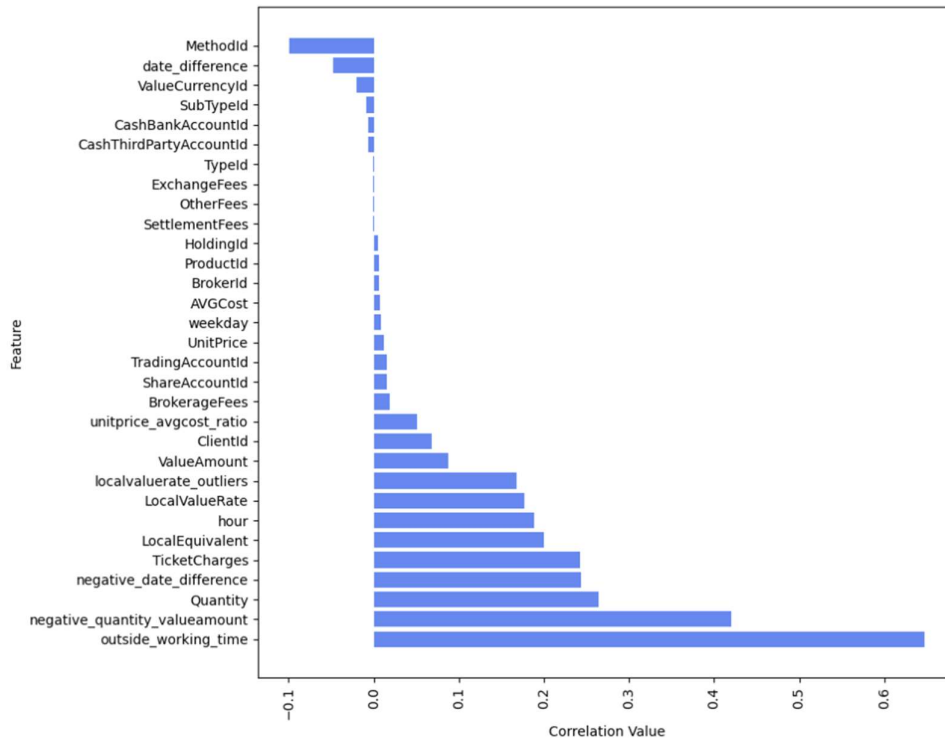
Anomaly types 4 and 6 are both point anomalies which required no feature extraction. That is because both types included records where values deviated significantly from the usual trend of one field. Hence, simple normalization techniques will do the job of detecting values that deviate from the normal trend.

**Table 6** shows a list of the new extracted features, which their definitions and data types.

**Table 6:** Name, Data Type and Description of New Extracted Fields

<i>New Feature Name</i>	<i>Data Type</i>	<i>Definition</i>
hour	int	Transaction hour (between 0 and 23)
weekday	int	Transaction weekday (Monday: 0, Tuesday: 1, Wednesday: 2, Thursday: 3, Friday: 4, Saturday: 5 and Sunday: 6)
date_difference	int	Difference between the share settlement date and the cash settlement date
outside_working_time	int	Binary field showing 1 if transactions are occurring outside of working days (all days except for Friday:4) and working hours (hour 8 till hour 18) and 0 otherwise
negative_date_difference	int	Binary field being 1 if the <i>date_difference</i> difference is negative, and zero otherwise
unitprice_avgcost_ratio	float	Ratio between unit price and the average price
localvaluerate_outliers	int	Binary field being 1 if the local settlement rate is an outlier relative to the usual local settlement rate for the same currency and 0 otherwise
negative_quantity_valueamount	int	Binary feature being 1 when quantity or value amount is negative, and 1 otherwise

To assess the impact of the newly extracted features, we generated a bar plot illustrating the correlations between the various features and the Anomaly label. The results indicate that the newly extracted features exhibit stronger correlations with the Anomaly label compared to a majority of the original features, as seen in **Figure 5**.

**Figure 5:** Anomaly Label Correlation with Different Features

### 3.4.3 Preprocessing Numerical Features

Before we fit our models, we needed to transform the features present in our datasets into values that bring out the best performance of models. First, we standardized the numerical features using the *StandardScaler* method from Scikit-learn library. This way, all the numerical features become on the same scale, ranging between -1 and 1. The formula for data normalization is presented below:

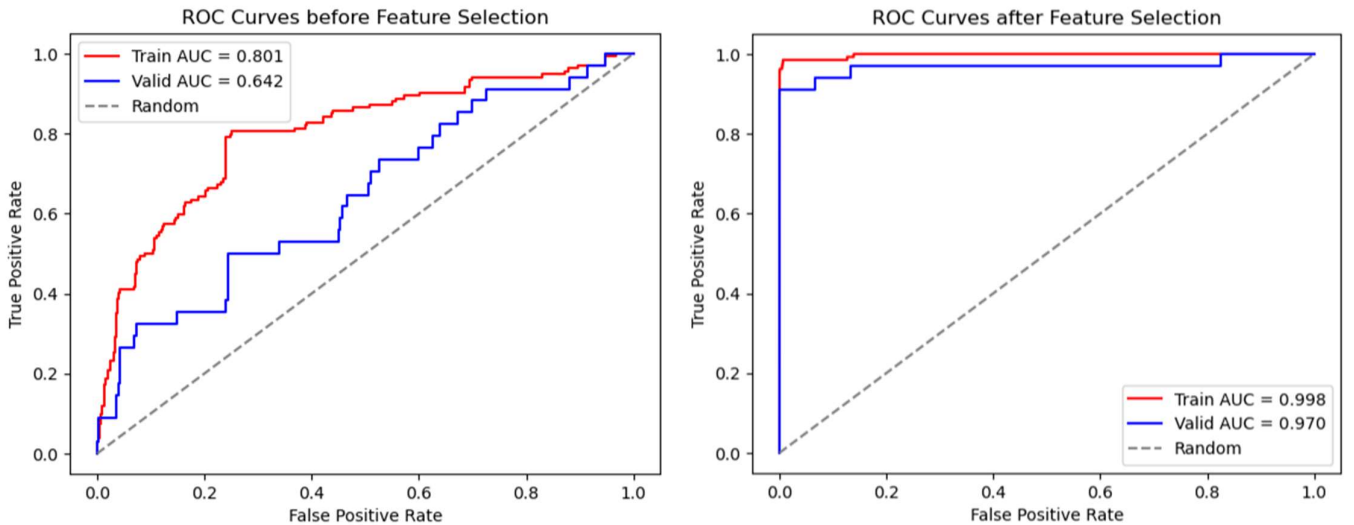
$$x_{std} = \frac{x - \mu}{\sigma} \quad (1)$$

Categorical features in our dataset are represented using numerical id features. These include *TypeId*, *SubTypeId*, *MethodId*, *HoldingId*, *ClientId*, *ProductId*, *BrokerId*, *TradingAccountId*, *ShareAccountId*, *CashBankAccountId*, and *ValueCurrenculd*. While they are present as numerical integers, they actually represent categorical entities. For example, the *TypeId* feature stands for the type of transaction, whether it is a sale or a purchase. We tried to apply one-hot-encoding for these categorical features. However, due to the many different unique values in each of these features, this technique led to a huge increase in the dimensionality of the dataset and resulted in a sparse dataset. Hence, we decided to keep these columns as they are, for they are already in numerical data types and will not impact the performance of our models.

### 3.4.4 Feature Selection

To identify the most relevant features, we employed the correlations between various features and the anomaly type feature. This approach ensures that features strongly associated with our target variable possess the highest predictive power. As seen in **Figure 7**, the most correlated features with the different anomaly types are: *Quantity*, *TicketCharges*, *ValueAmount*, *LocalValueRate*, *LocalEquivalent*, *outside\_working\_time*, *negative\_date\_difference*, *unitprice\_avgcost\_ratio*, *localvaluerate\_outliers*, and *negative\_quantity\_valueamount*.

To evaluate the effects of feature selection, we conducted two rounds of logistic regression training. The first iteration involved all features, while the second iteration employed only the selected features. As depicted in the ROC Curves of **Figure 6**, this selection process led to substantial enhancement in performance, elevating the AUC score on the validation dataset from 0.642 to an impressive 0.97.



**Figure 6:** ROC Curves of Logistic Regression before and after Feature Selection

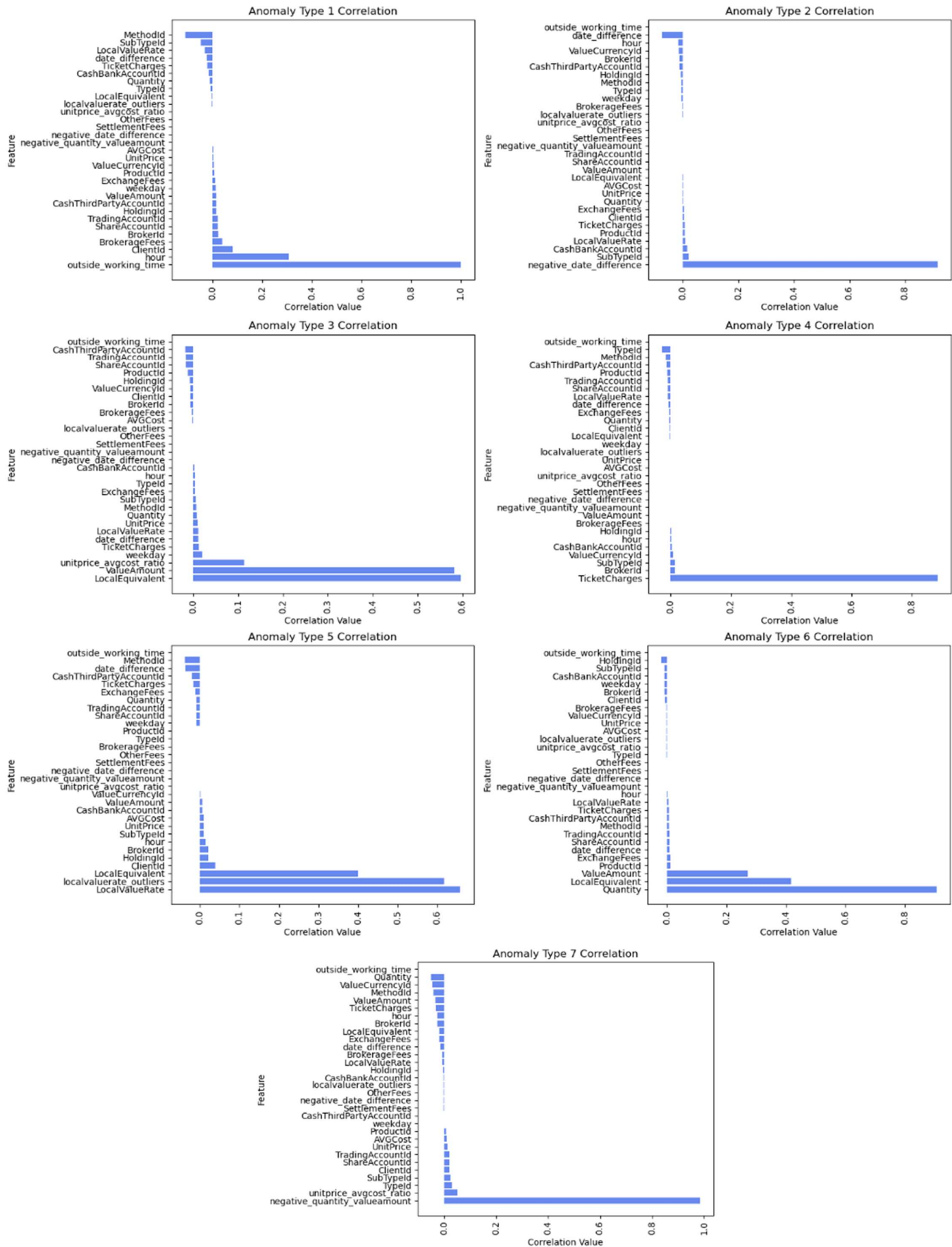


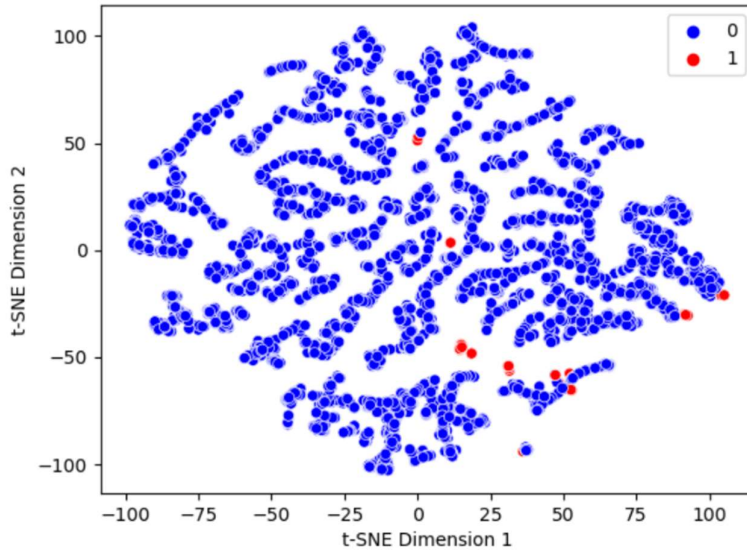
Figure 7: Correlation of Anomaly Type with the Different Features

### 3.5 Modelling and Fine-Tuning

Given the fact that our data is now labeled, we decided to train both supervised and unsupervised models. For supervised models, we trained the models based on the binary anomaly labels. The unsupervised models only used the labels for evaluation purposes. All in all, we trained eight types of supervised models and four types of unsupervised models. In this section, we describe the modelling steps, including the train and test splitting, evaluation metrics used, and the different models that were trained.

#### 3.5.1 Data Visualization using t-SNE Plot

Before we started our modelling phase, we decided to visualize our data in a two-dimensional plot in order to better understand our anomalies and refine our modelling approaches accordingly. The t-Distributed Stochastic Neighbor Embedding (t-SNE) method serves as a valuable tool to transform the data into a two-dimensional shape, allowing us to plot the data, as displayed in **Figure 8**.



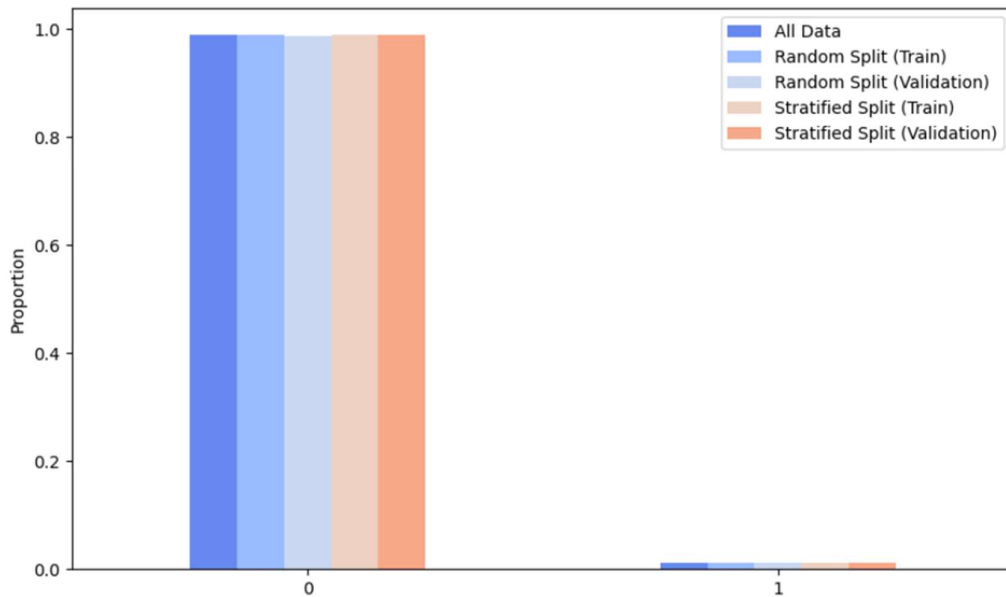
**Figure 8:** t-SNE Scatter Plot

Our analysis of the t-SNE visualization reveals the presence of distinct clusters that correspond to anomalous data points. These clusters highlight instances where anomalies exhibit a discernible separation from the bulk of normal data points. They are either distant from normal points, gathered in an almost empty space, or on the edges of normal data. This observation reaffirms the effectiveness of t-SNE in detecting groups of anomalies that share similar patterns.

However, it's worth noting that some anomalies exhibit a degree of overlap with the normal data points, which could potentially complicate their detection. Instances of such overlap challenge our understanding of the dataset's underlying structure and underscore the complexity of the anomaly detection task. This observation underscores the importance of employing complementary techniques and methodologies to distinguish anomalies that are subtly interwoven with the normal data. As such, we decided to experiment with a substantial number and types of models in order to capture the biggest number of these nuances.

### 3.5.2 Train-Test Split

We started our modelling with optimizing the splitting into training and testing datasets. Knowing that the data labels are highly imbalanced, we compared the proportions of both labels between the original data, a randomly split data and a stratified split data. The results are shown in **Figure 9**.



**Figure 9:** Bar Plot of the Comparison of Split Proportions

As seen in the figure above, proportion of normal and anomalous data in the original dataset, the randomly split training dataset, the randomly split validation dataset, the stratified split training dataset and the stratified split validation dataset are all almost equal. So, it will not affect the results if we used any of the splitting techniques. To stay on the safe side, we decided to implement a stratified split for training our models.

It is important to note here that all the previously described data preparation steps (cleaning, feature extraction, and scaling) were all fitted to the training dataset and then transformed onto the training and validation datasets. The main reason for that is to avoid leakage of data, which would cause overfitting of the models.

### 3.5.3 Evaluation Metrics

In the modeling phase of our analysis, the assessment of the performance of our anomaly detection model is a critical step to ensure its effectiveness. A wide range of evaluation metrics is available to measure the model's performance in different aspects. The choice of evaluation metrics is influenced by the nature of the problem, the characteristics of the dataset, and the specific goals of the analysis.

Several evaluation metrics are commonly used in anomaly detection tasks, each providing insights into different aspects of the model's performance:

- *Accuracy*: This metric measures the proportion of correctly predicted instances (both anomalies and normal) among all instances. However, accuracy can be misleading in the context of imbalanced datasets, such as ours, where anomalies represent only a small fraction of the data. A high accuracy score can be achieved by a model that merely labels all instances as normal due to the overwhelming prevalence of normal instances.
- *Precision and Recall*: Precision is the ratio of true positive predictions to the total predicted positives  $\left(\frac{TP}{TP + FP}\right)$ , while recall is the ratio of true positive predictions to the total actual positives  $\left(\frac{TP}{TP + FN}\right)$ . Precision emphasizes the proportion of true anomalies among the instances predicted as anomalies, while recall focuses on the model's ability to capture actual anomalies.
- *F1-Score*: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance on both false positives and false negatives.
- *Area Under the Receiver Operating Characteristic Curve (AUC-ROC)*: The AUC-ROC score evaluates the model's ability to distinguish between anomaly and normal instances by plotting the true positive rate against the false positive rate across different classification thresholds.

Given the highly imbalanced nature of our dataset (99% normal instances and only 1% anomalies), we chose to focus on evaluation metrics that are better suited for imbalanced scenarios. Specifically, we employed True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Macro-averaged recall, and the AUC score.

The choice to avoid accuracy as the primary metric is motivated by the fact that an accuracy score could be misleading due to the imbalanced distribution of classes. For instance, if the model naively labels all instances as normal, it would still achieve a high accuracy of 99%, despite failing to detect any true anomalies. By using recall and AUC, we prioritize the model's ability to identify anomalies accurately, thus mitigating the impact of the class imbalance.

### 3.5.4 Supervised Learning

In our pursuit of building an effective anomaly detection system, we harnessed the capabilities of a diverse set of supervised learning models. These models were chosen to cover a broad spectrum of techniques, encompassing both traditional machine learning (ML) approaches and a deep learning (DL) architecture. This subsection will provide a comprehensive overview of the models used, their descriptions, and the fine-tuning strategies employed to enhance their performance. The definitions provided below are stated in the official documentations of *Scikit-learn*, *XGboost*, *TensorFlow* and *Keras*.

Supervised Machine Learning Techniques by *Scikit-learn*:

#### ○ Logistic Regression:

The Logistic Regression is a classical linear classification algorithm that estimates the probability that an instance belongs to a particular class. Despite its simplicity, it serves as a solid baseline model for binary classification tasks like anomaly detection due to its probabilistic and low-processing nature.

The likelihood of the outcome variable  $y$  being 1, when considering a single input variable  $x$ , is determined using the equation:

$$P(y = 1|x) = \frac{e^{a+bx}}{1 + e^{a+bx}} = \frac{1}{1 + e^{-(a+bx)}} \quad (2)$$



This equation represents the logistic function, where the parameters  $a$  and  $b$  determine the slope of the function. (Peng et. al., 2002)

- K-Nearest Neighbors (KNN):

The KNN model is an instance-based algorithm that classifies a data instance based on the majority class among its  $k$ -nearest neighbor points. It captures local patterns in the data and can be effective in cases where anomalies form distinct clusters (Sutton, O., 2012). Some commonly used equations for measuring the distance between datapoints are the Manhattan and the Euclidean distances.

- Gaussian Naïve Bayes:

The Gaussian Naïve Bayes model is a probabilistic algorithm that assumes that features are independent given a class and that the continuous features follow a Gaussian distribution. It works well when the data distribution aligns with its assumptions and is computationally efficient (Rish I., 2001). Using Bayes' theorem, the calculation of a posterior probability is expressed as follows:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (3)$$

This equation is employed to compute probabilities and select the highest one for each class  $c$ . The likelihood assesses how likely the evidence is for a particular class  $c$ , while the class prior probability represents the initial likelihood of the class before considering the evidence. The predictor prior probability refers to the overall likelihood of the evidence  $x$  across all possible classes (Rish I., 2001).

- Support Vector Machine (SVM):

The Support Vector Machines model provided successful results in a wide range of classification problems, which made us consider testing it on our anomaly detector. SVM is a powerful algorithm that aims to find an optimal hyperplane that maximally separates classes. It works well in high-dimensional spaces and can capture complex relationships between features (Bakumenko & Alregal, 2022). The decision function of a Support Vector Machine (SVM) is presented as follows:

$$F(x) = \sum_i^N \alpha_i y_i k(x_i, x) + b \quad (4)$$

In this equation, for each training sample,  $b$  represents a bias term,  $y_i$  is the desired output for that sample,  $\alpha_i$  is a weight factor, and  $k(x_i, x)$  signifies the kernel function applied to the support vector  $x_i$  (Bakumenko & Alregal, 2022).

- Decision Tree:

A Decision Tree model builds a tree-like structure to make decisions by repeatedly splitting the data based on the most discriminative features. It is intuitive, interpretable, and can capture non-linear relationships (Jijo & Abdulazeez, 2021).

- Random Forest:

The Random Forest is one of the best performing models for supervised anomaly detection and classification models. It is an ensemble technique that aggregates the predictions of multiple

decision trees through the use of bootstrapped splitting of the data. It reduces overfitting and enhances robustness by averaging the outcomes of various trees (Bakumenko & Alregal, 2022).

- Extreme Gradient Boosting (XG-Boost):  
XG-Boost is a boosting algorithm that sequentially builds weak models and combines their predictions to form a strong model. It excels in capturing complex patterns and handling imbalanced data. It also has a high speed as it is executed through parallel computation. (Brownlee, 2016)

Supervised Deep Learning techniques using *TensorFlow*:

- Artificial Neural Networks (ANN):  
Our DL approach utilizes the ANN architecture from the *TensorFlow's Keras* library. ANNs consist of interconnected nodes, or neurons, organized in layers. By varying the number of layers and neurons, ANNs can capture intricate patterns in the data. According to the description given by Bakumenko and Alregal (2022), the common arrangement in neural networks involves layers of neurons, with the initial layer receiving input features and the final layer producing the model's output. Between these layers, there is a hidden group of neurons. In a fully connected feedforward NN, every neuron in one layer is linked to each neuron in the subsequent layer. The neuron weights are modified using the backpropagation technique, which involves comparing desired and predicted outputs. Through this iterative weight adjustment process, a neural network can learn from training data.

We fine-tuned the ML models using *GridSearch* from *Scikit-learn*, systematically exploring hyperparameter combinations to optimize performance by maximizing the AUC score on the validation dataset.

For the ANN models, we experimented with three configurations: one hidden layer, two hidden layers, and three hidden layers. In each of these ANN models, the *ReLU* activation function was used for the hidden layers and the *Sigmoid* for the output layer since our output is binary. We employed the *RandomSearch* technique from *KerasTuner* to identify optimal hyperparameters for the ANN models also by maximizing the AUC score.

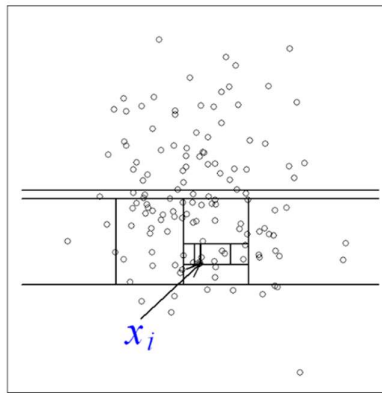
### 3.5.5 Unsupervised Learning

In our attempt to construct a robust anomaly detection system, we delved into the realm of unsupervised learning models. These models, which operate without labeled data, are proficient at uncovering patterns and deviations inherent within the data itself. This methodology section aims to provide a comprehensive overview of the unsupervised learning models employed, their descriptions, and the strategies we employed to fine-tune their detection capabilities to cater to the unique anomalies present in our dataset. The below definitions are mentioned in the documentations of *PyOD* and *PyTorch* libraries.

## Unsupervised Machine Learning Techniques by *PyOD*:

- Isolation Forest:

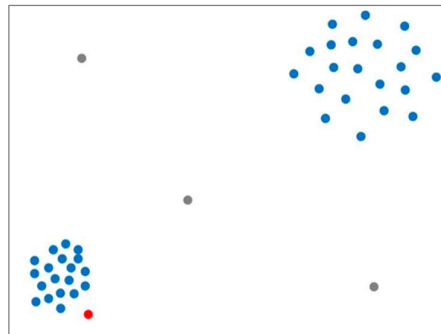
The Isolation Forest algorithm is an ensemble-based approach that isolates anomalies by partitioning the data into subsets. It effectively measures the average depth at which data points are isolated, with anomalies tending to have shorter average path lengths. This approach encompasses two parameters for training and one parameter for evaluation. The training parameters consist of the count of constructed trees and the size of subsampling. The evaluation parameter pertains to the maximum allowed height of trees during the evaluation process (Liu, F. T. et. al, 2012). **Figure 10** illustrated in the paper of Liu et al. (2012) shows the isolation of a data point  $x_i$  in a two-dimensional space.



**Figure 10:** Isolating  $x_i$ . Adapted from Liu F. T. et. al., 2012

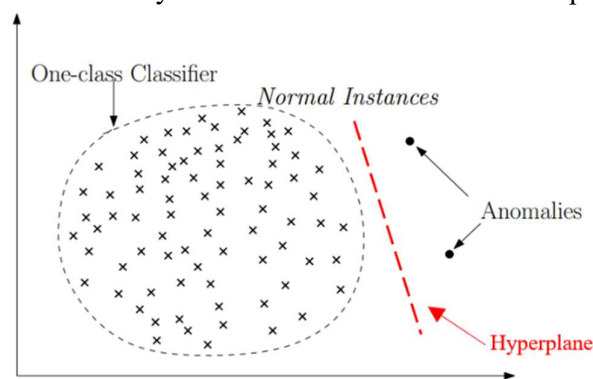
- Local Outlier Factor (LOF):

The Local Outlier Factor is another commonly used unsupervised machine learning model for anomaly or outlier detection. According to a paper written by Alghushairy et al. (2020), this algorithm measures the local density deviation of data points compared to their neighbors using the KNN model. Anomalies exhibit lower neighborhood density, resulting in higher LOF scores. **Figure 11** shows the two different types of outliers, where the red point is a local outlier and the grey is the global outlier.



**Figure 11:** Different types of outliers. Adapted from Alghushairy et al., 2020

- One-Class Support Vector Machine (One-Class SVM):  
The One-Class SVM is based on the previously described SVM model. This model constructs a hyperplane that captures the majority of the data points, treating points outside the hyperplane as anomalies. It is particularly suited for datasets with a small proportion of anomalies just like our dataset. According to Chandola et al. (2009), the algorithm identifies the narrowest hyperplane within the kernel space that encompasses all the training examples. It then establishes the relative position of a test instance in relation to this hypersphere. If the test instance falls outside the hypersphere, it is categorized as an anomaly. **Figure 12** depicts an illustration of one-class anomaly detection in a two-dimensional space.

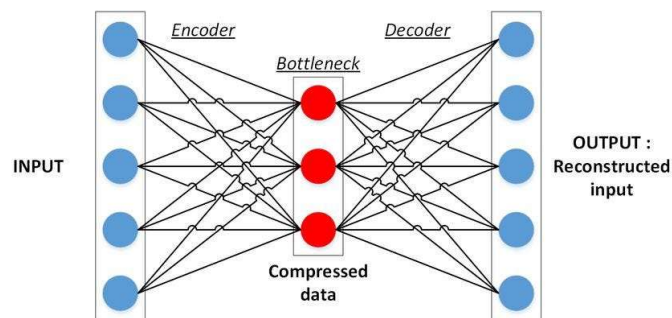


**Figure 12:** One-Class SVM Anomaly Detection. Adapted from Chandola et al., 2009

We fine-tuned these unsupervised learning models using *GridSearch* from Scikit-learn, configuring their parameters to achieve the highest AUC score. This process ensured that the models were optimized to capture the unique characteristics of our anomalies.

Unsupervised Deep Learning techniques using *TensorFlow*:

- Auto-Encoder Model:  
An Auto-Encoder consists of an encoder network that compresses only the normal input data into a lower-dimensional latent space and a decoder network that reconstructs the original normal input from the compressed representation, as seen in **Figure 13** (Pang et. al., 2021). This way, the model is trained to understand the normal state of the data, thus identifying any deviation from the normal as anomalous.



**Figure 13:** Basic Architecture of an AutoEncoder Model. Adapted from Sublime and Kalinicheva, 2019

According to a study by Pang et al. (2021), Autoencoder-driven anomaly detection has the capability to tackle four of the six primary challenges associated with anomaly detection. These challenges encompass issues such as achieving a low recall rate in anomaly detection, dealing with a large number of features, handling noisy data, and managing intricate relationships between features. However, there are certain limitations to autoencoders, including their reliance on the amount of data and their inability to provide interpretations.

The Auto-Encoder model was fine-tuned by manually adjusting the loss threshold. Through iterative experimentation, we identified the threshold that yielded the best performance in detecting anomalies.

In the subsequent **Results** section, we will delve into the outcomes of the hyperparameter tuning and in the **Discussion** section, we tackle the comparative performance analysis of all the models described above, highlighting their contributions toward detecting anomalies effectively within our dataset.

### 3.6 Evaluation

In the evaluation phase, the trained models went through a thorough evaluation to determine their performance on previously unseen data. After training and tuning the parameter for each model, the models were evaluated using the validation dataset. The validation dataset, which was held out during the training process, allowed us to simulate the real-world scenario of encountering new data instances.

To quantitatively evaluate the models, several key metrics were employed, including the Receiver Operating Characteristic (ROC) curves, confusion matrices, and classification reports. ROC curves provide insights into the trade-off between the true positive rate and the false positive rate for varying classification thresholds. The confusion matrices offered a detailed breakdown of the predictions, showcasing the number of true positives, true negatives, false positives, and false negatives. The classification reports presented a comprehensive summary of precision, recall, F1-score, and support for each class.

By comparing the results obtained from these evaluation metrics across the various models, it was possible to discern the strengths and weaknesses of each approach. This rigorous evaluation process facilitated the identification of the best performing model among the alternatives, ensuring that the chosen model demonstrated consistent and robust performance across different metrics. The outcomes of this evaluation will be presented in the **Results** section.

### 3.7 Deployment

After selecting the best-performing model, the chosen models were deployed on a *Streamlit* platform, providing a user-friendly interface for interaction. Prior to deployment, appropriate preprocessing steps were applied to incoming data to ensure compatibility with the model's requirements.

For efficient deployment, both the preprocessing steps and the trained models were serialized and saved using the *pickle* library. This allowed the models to be reconstructed with their learned parameters intact, ensuring that the deployment experience mirrors the training process as closely as possible. The deployment code was written in a separate Python file specific for integration with the *Streamlit* framework.

Screenshots of the *Streamlit* application will be shown in the **Results** section as well.

# 4 Results

In this section, we present the results of the hyperparameter tuning process and the evaluation of each implemented model on the validation dataset. For each model, we provide the Receiver Operating Characteristic (ROC) curve, confusion matrix, and classification report to comprehensively assess its performance.

## 4.1 Supervised Learning

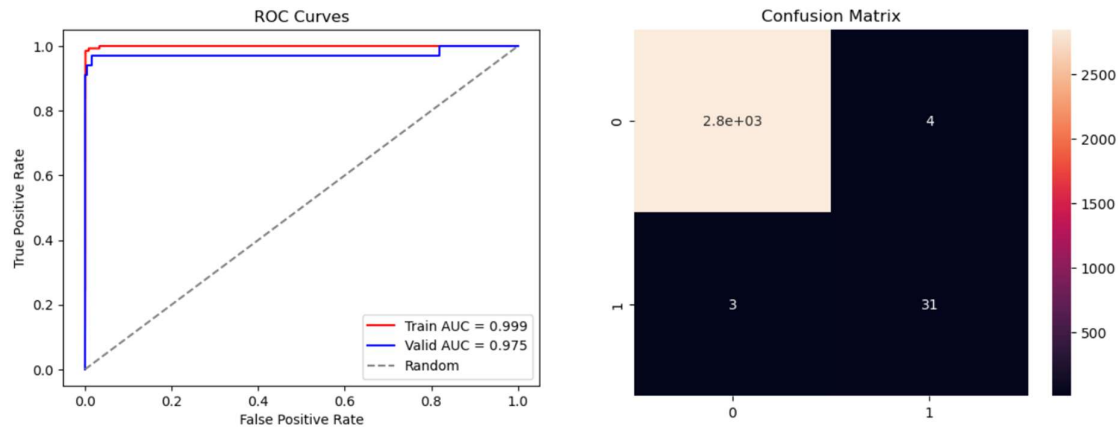
The supervised models were trained on the selected features and fine-tuned to enhance their performance.

### Logistic Regression

After tuning the hyperparameters of the Logistic Regression model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as  $C=1$ , *class\_weight* as *balanced*, and a *maximum iteration limit* of 100. **Table 7** shows the Logistic Regression's Classification Report and **Figure 14** shows the Logistic Regression's ROC Curve and Confusion Matrix.

**Table 7:** Logistic Regression Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	0.89	0.91	0.90	34
Accuracy			1.00	2888
Macro Avg	0.94	0.96	0.95	2888
Weighted Avg	1.00	1.00	1.00	2888



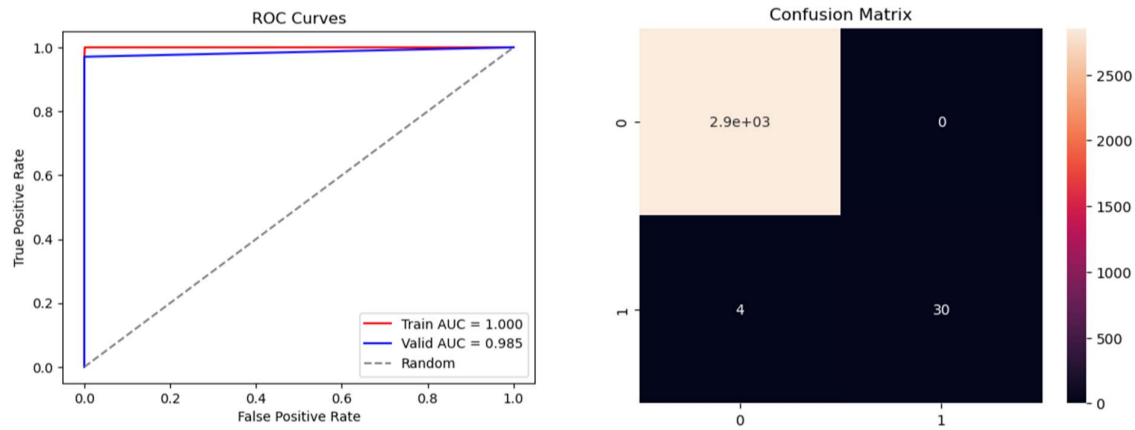
**Figure 14:** Logistic Regression ROC Curves and Confusion Matrix

## K-Nearest Neighbors

After tuning the hyperparameters of the K-Nearest Neighbors model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as *n\_neighbors* set to 5 and *weights* set to *uniform*. **Table 8** shows the KNN model's Classification Report and **Figure 15** shows the model's ROC Curve and Confusion Matrix.

**Table 8:** KNN Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	1.00	0.88	0.94	34
Accuracy			1.00	2888
Macro Avg	1.00	0.94	0.97	2888
Weighted Avg	1.00	1.00	1.00	2888



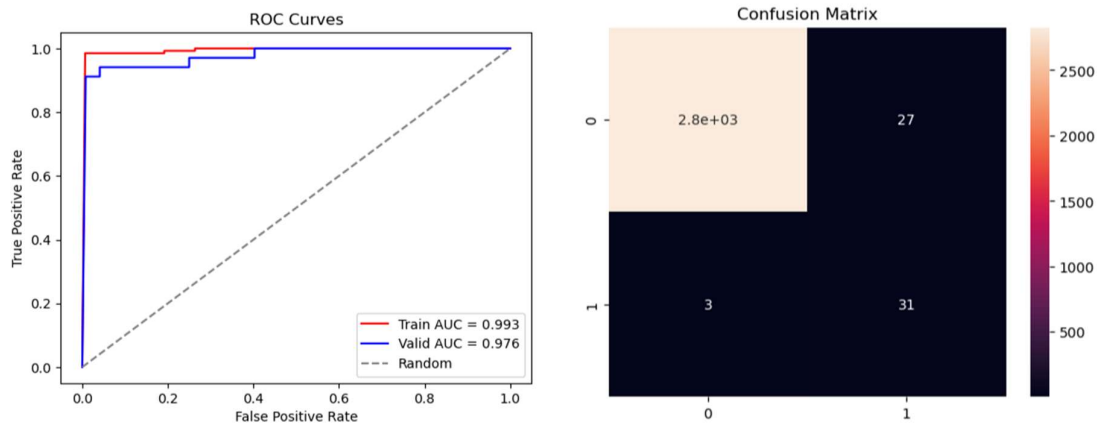
**Figure 15:** KNN ROC Curves and Confusion Matrix

## Gaussian Naïve Bayes

After tuning the hyperparameters of the Naïve Bayes model, the optimal hyperparameter, which resulted in the maximum AUC score, were identified as *var\_smoothing* set to *1e-09*. **Table 9** shows the Naïve Bayes model's Classification Report and **Figure 16** shows the model's ROC Curve and Confusion Matrix.

**Table 9:** Naïve Bayes Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	0.99	0.99	2854
1	0.53	0.91	0.67	34
Accuracy			0.99	2888
Macro Avg	0.77	0.95	0.83	2888
Weighted Avg	0.99	0.99	0.99	2888



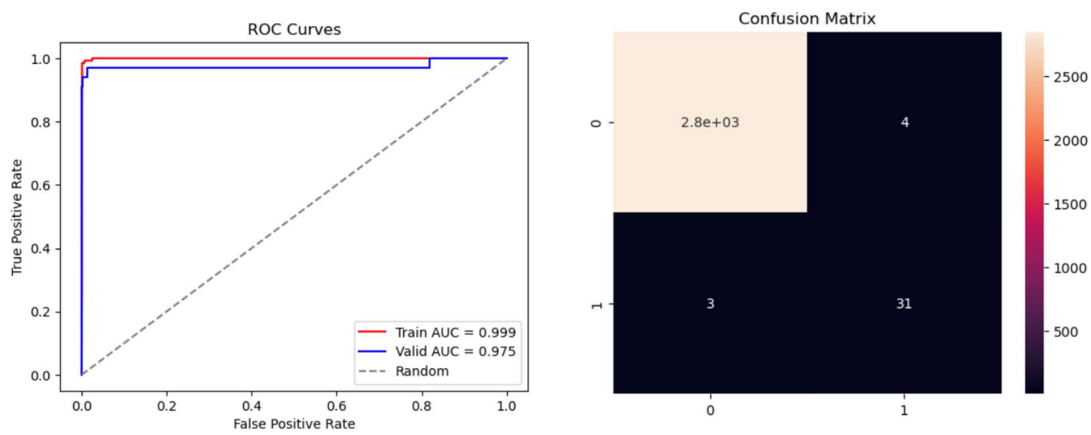
**Figure 16:** Naïve Bayes ROC Curves and Confusion Matrix

## Support Vector Machine

After tuning the hyperparameters of the SVM model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as  $C$  set to 0.1,  $class\_weight$  set to *balanced*,  $gamma$  being *scale* and the  $kernel$  being *linear*. **Table 10** shows the Naïve Bayes model's Classification Report and **Figure 17** shows the model's ROC Curve and Confusion Matrix.

**Table 10:** SVM Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	0.89	0.91	0.90	34
Accuracy			1.00	2888
Macro Avg	0.94	0.96	0.95	2888
Weighted Avg	1.00	1.00	1.00	2888



**Figure 17:** SVM ROC Curves and Confusion Matrix

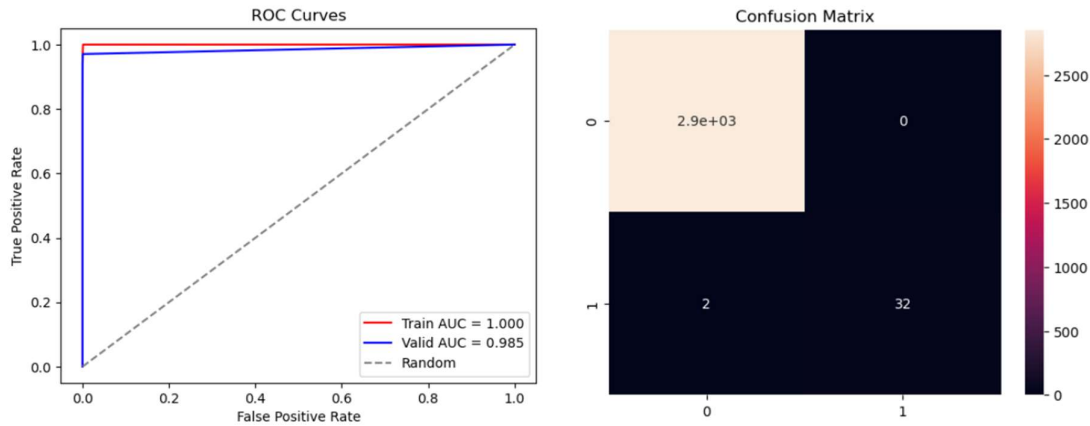


## Decision Tree

After tuning the hyperparameters of the Decision Tree model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as *class\_weight* set to None, *criterion* being *gini*, *max\_depth* set to None, *min\_samples\_leaf* set to 2, *min\_samples\_split* set to 10 and *splitter* set to *random*. **Table 11** shows the Decision Tree's Classification Report and **Figure 18** shows the model's ROC Curve and Confusion Matrix.

**Table 11:** Decision Tree Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	1.00	0.94	0.97	34
Accuracy			1.00	2888
Macro Avg	1.00	0.97	0.98	2888
Weighted Avg	1.00	1.00	1.00	2888



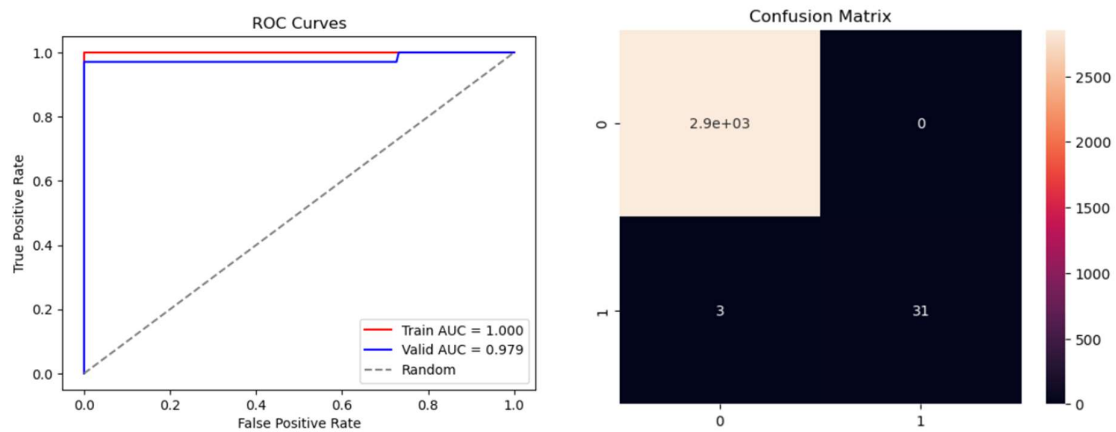
**Figure 18:** Decision Tree ROC Curves and Confusion Matrix

## Random Forest

After tuning the hyperparameters of the Random Forest model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as *class\_weight* set to None, *max\_depth* set to None, *min\_samples\_leaf* set to 1, *min\_samples\_split* set to 10 and *n\_estimators* set to 100. **Table 12** shows the Random Forest's Classification Report and **Figure 19** shows the model's ROC Curve and Confusion Matrix.

**Table 12:** Random Forest Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	1.00	0.91	0.95	34
Accuracy			1.00	2888
Macro Avg	1.00	0.96	0.98	2888
Weighted Avg	1.00	1.00	1.00	2888



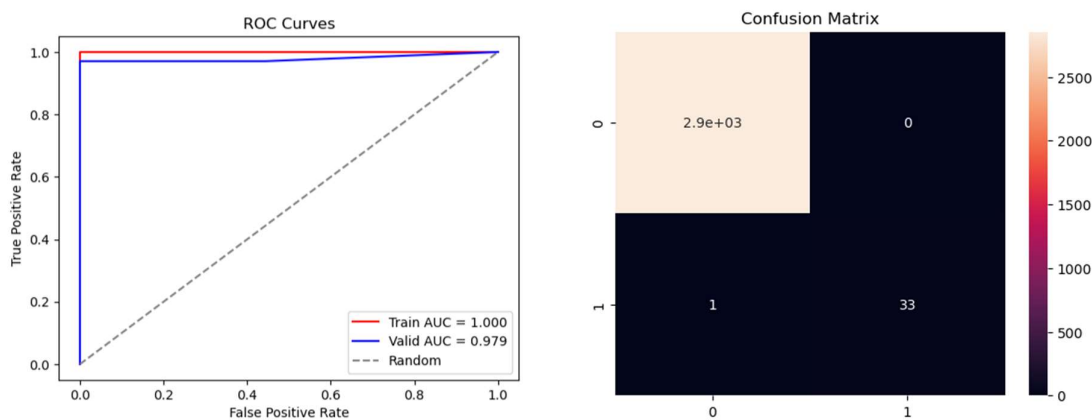
**Figure 19:** Random Forest ROC Curves and Confusion Matrix

## Extreme Gradient Boosting

After tuning the hyperparameters of the XGBoost model, the optimal hyperparameters, which resulted in the maximum AUC score, were identified as *colsample\_bytree* set to 0.8, *learning\_rate* set to 0.1, *max\_depth* set to 3, *n\_estimators* set to 100, *scale\_pos\_weight* set to 5, and *subsample* set to 0.8. **Table 13** shows the XGBoost's Classification Report and **Figure 20** shows the model's ROC Curve and Confusion Matrix.

**Table 13:** XGBoost Classification Report

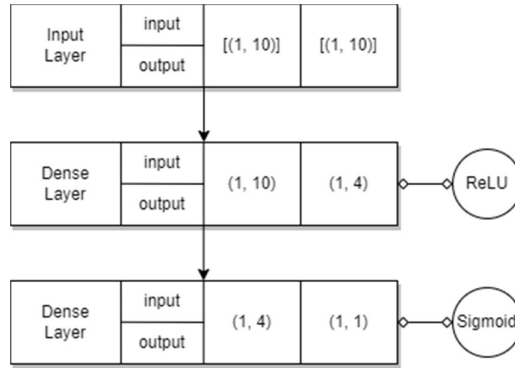
	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	1.00	0.97	0.99	34
Accuracy			1.00	2888
Macro Avg	1.00	0.99	0.99	2888
Weighted Avg	1.00	1.00	1.00	2888



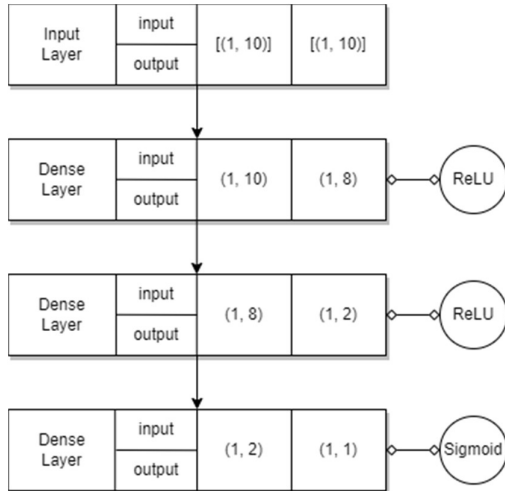
**Figure 20:** XGBoost ROC Curves and Confusion Matrix

## Artificial Neural Networks

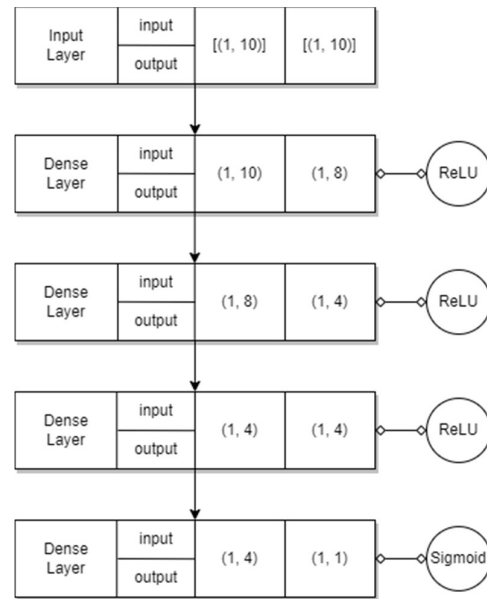
We trained and fine-tuned three deep neural networks, iteratively increasing the number of hidden layers. The three best-performing ANNs are presented in the diagrams of **Figures 21, 22** and **23**. The models were trained for 50 epochs using *Binary\_crossentropy* loss and *adam* optimizer with a *learning\_rate* of 0.01.



**Figure 21:** Architecture of ANN1

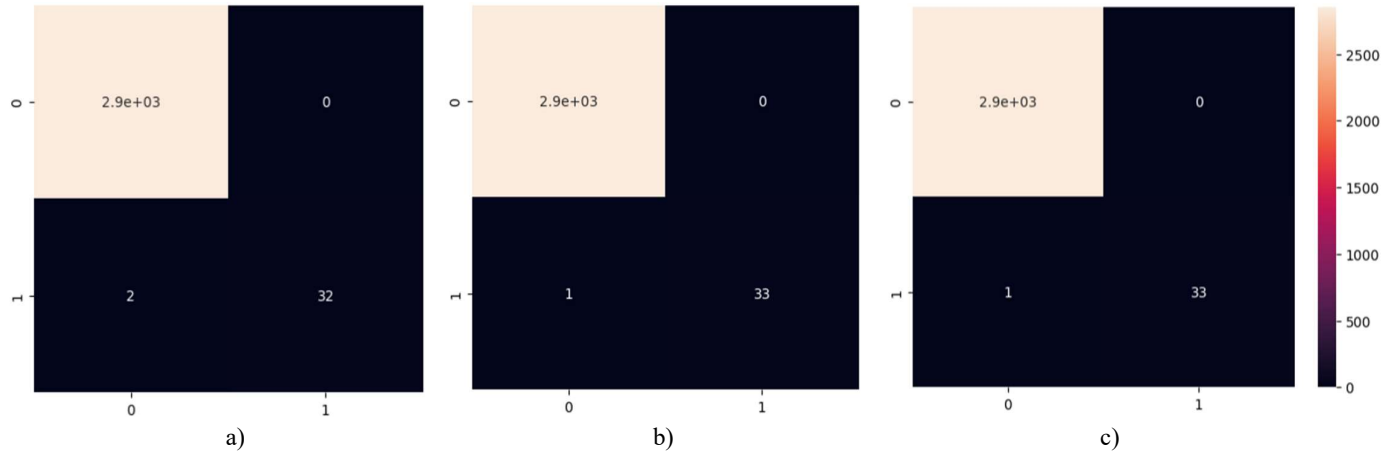


**Figure 22:** Architecture of ANN2

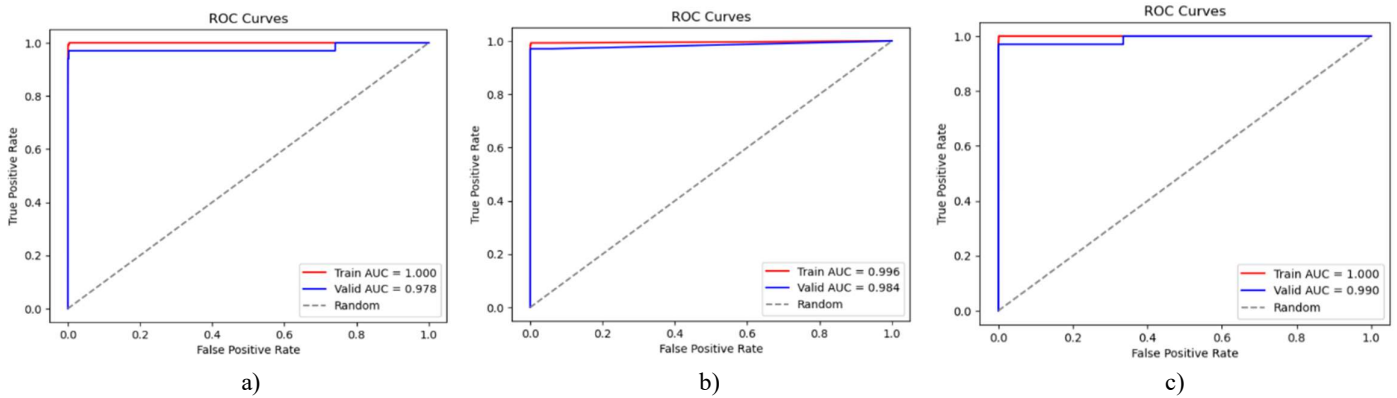


**Figure 23:** Architecture of ANN3

All three models showed similar performances. The confusion matrices and ROC Curves of the three models are presented in **Figure 24** and **Figure 25**, respectively.



**Figure 24:** Confusion matrices of the three artificial neural networks, a) ANN1, b) ANN2, c) ANN3



**Figure 25:** ROC Curves of the three artificial neural networks, a) ANN1, b) ANN2, c) ANN3

The three models performed very similarly. Even though ANN2 and ANN3 have the same confusion matrices, their ROC curves and AUC scores are slightly different, with ANN3 scoring a 0.99 AUC on the validation dataset. They were able to capture 33 out of 34 anomalies in the validation dataset. **Table 14** shows the Classification Report of the third ANN model.

**Table 14:** ANN3 Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	1.00	1.00	2854
1	1.00	0.97	0.99	34
Accuracy			1.00	2888
Macro Avg	1.00	0.99	0.99	2888
Weighted Avg	1.00	1.00	1.00	2888

## 4.2 Unsupervised Learning

The unsupervised models were trained on the selected features. We wanted to evaluate the performance of the models on the specific types of anomalies that we were interested in detecting, and so the selected features will help in improving their predictive power. The models were also fine-tuned based on their ability to detect these specific anomalies by maximizing the AUC score.

### Isolation Forest

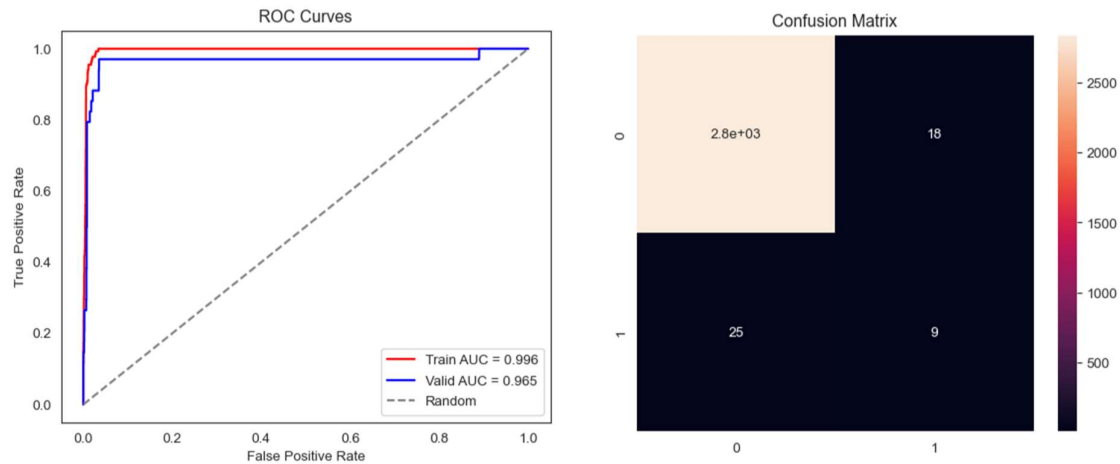
The optimal hyperparameters of the Isolation Forest are presented below:

- *contamination*: This parameter represents the proportion of outliers in the data set. The best value turned out to be 0.01, which means that the model is expected to find 1% of outliers in the data set.
- *max\_samples*: This parameter controls the maximum number of samples that will be used to train each tree. The best value turned out to be 0.1, which means that 10% of the samples will be used to train each tree.
- *n\_estimators*: This parameter represents the number of trees in the forest. The best value turned out to be 100.

**Table 15** shows the model's Classification Report and **Figure 26** shows the model's ROC Curve and Confusion Matrix.

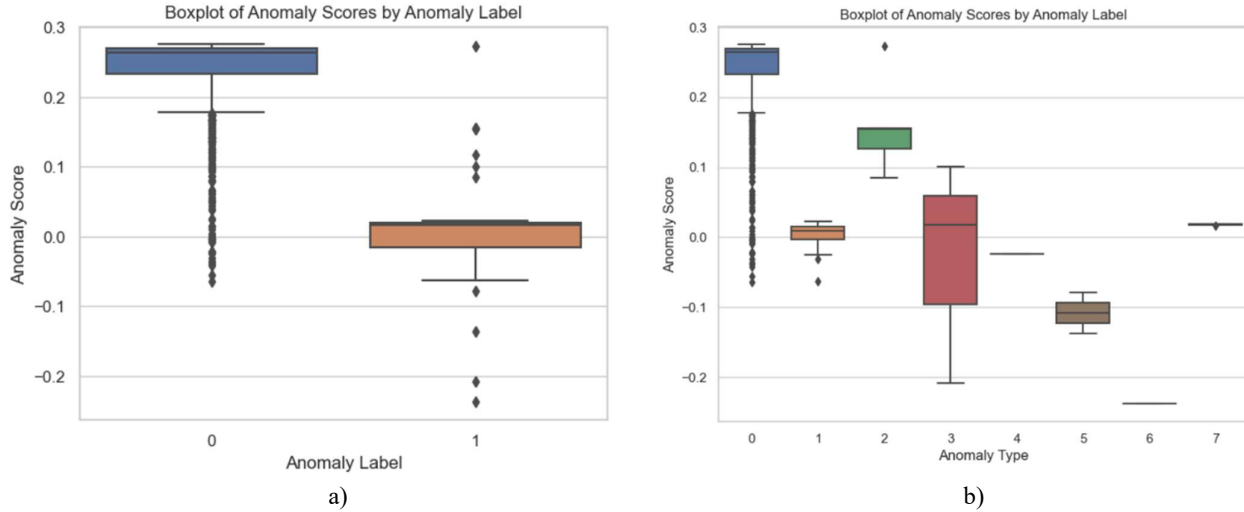
**Table 15:** Isolation Forest Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	0.99	0.99	0.99	2854
1	0.33	0.26	0.30	34
Accuracy			0.99	2888
Macro Avg	0.66	0.63	0.64	2888
Weighted Avg	0.98	0.99	0.98	2888



**Figure 26:** Isolation Forest ROC Curves and Confusion Matrix

The *Decision\_function* method by *PyOD library* estimates anomaly scores based on the contamination parameter. It measures how likely a data point is to be an anomaly. **Figure 27** shows the model's anomaly scores for, **(a)** each actual anomaly binary class and **(b)** each anomaly type. The lower the anomaly score is, the higher is the likelihood of the data point being an anomaly. As seen in the figure, the model is able to identify the different type.



**Figure 27:** Isolation Forest Anomaly Score Boxplots per **a)** Anomaly Class and **b)** Anomaly Type

## Local Outlier Factor

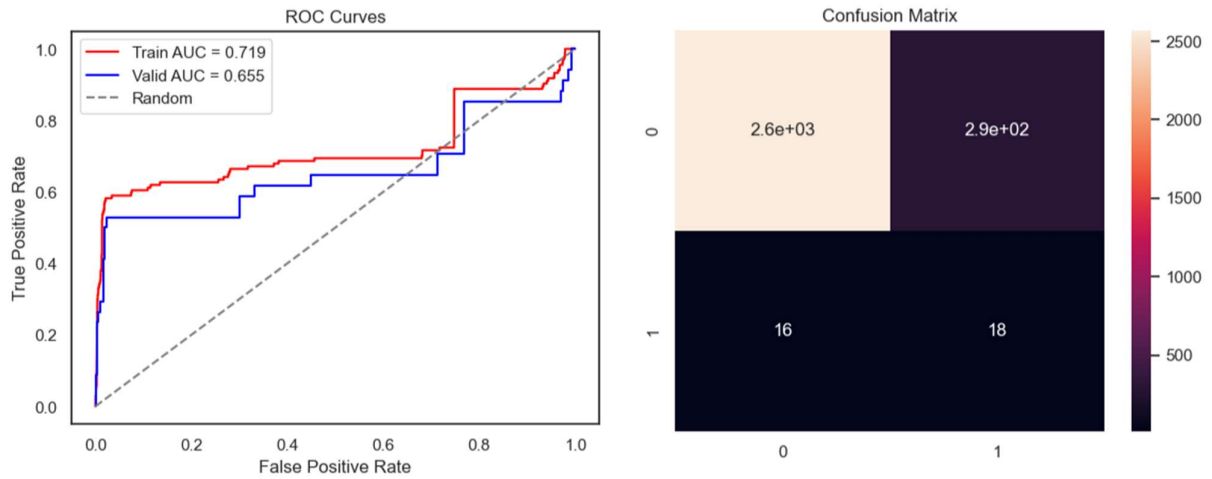
The optimal hyperparameters of the Local Outlier Factor model are presented below:

- *contamination*: This parameter represents the proportion of outliers in the data set. The best value was 0.1, which means that the model is expected to find 10% of outliers in the data set.
- *n\_neighbors*: This parameter controls the number of neighbors that will be used to calculate the local density of each data point. The best value turned out to be 20, meaning that 20 neighboring data points for every point will be used to calculate the local density.

**Table 16** shows the model's Classification Report and **Figure 28** shows the model's ROC Curve and Confusion Matrix.

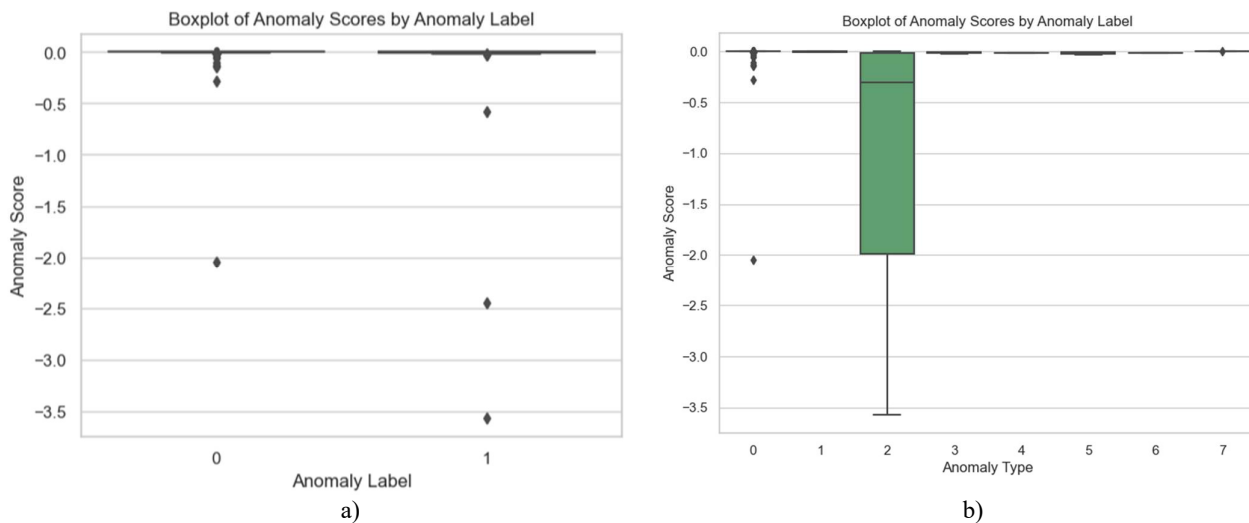
**Table 16:** Local Outlier Factor Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	0.99	0.90	0.94	2854
1	0.06	0.53	0.11	34
Accuracy			0.90	2888
Macro Avg	0.53	0.71	0.53	2888
Weighted Avg	0.98	0.90	0.93	2888



**Figure 28:** Local Outlier Factor ROC Curves and Confusion Matrix

**Figure 29** shows the boxplots of the anomaly scores per anomaly class and per anomaly type. The model seems to be weak at distinguishing between the different normal and anomalous data points due to the similarity between plots of the different classes and types. It worked relatively better at capturing anomalies of type 2 than any other type.



**Figure 29:** Local Outlier Factor Anomaly Score Boxplots per **a)** Anomaly Class and **b)** Anomaly Type

## One-Class Support Vector Machine

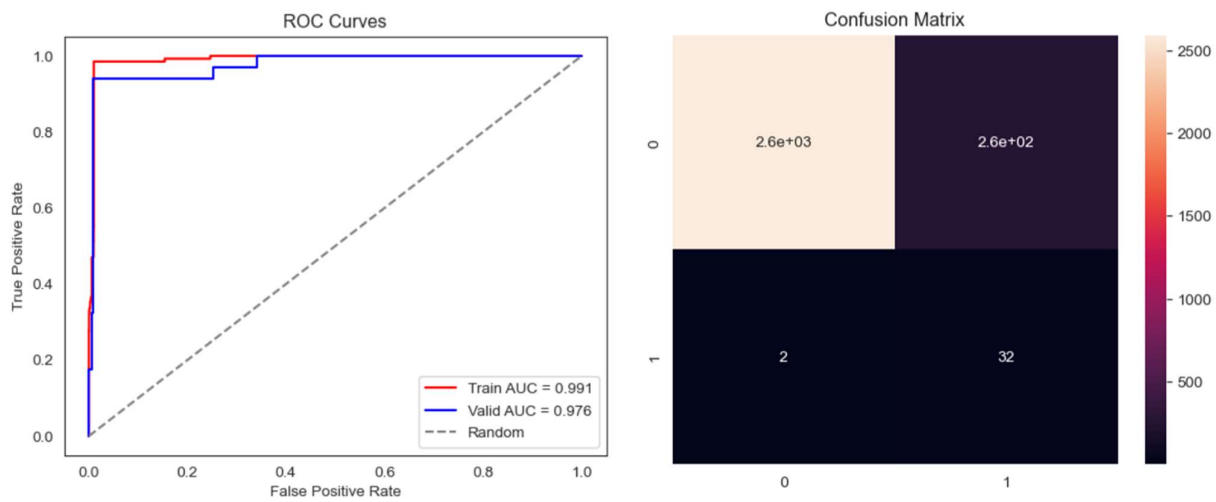
The optimal hyperparameters of the One-Class SVM model are presented below:

- *contamination*: The optimal value was 0.1, which means that the model is expected to find 10% of outliers in the data set.
- *gamma*: This parameter controls the kernel width of the Gaussian kernel. A higher value will result in a narrower kernel, which can lead to more data points being classified as outliers. The optimal value turned out to be "auto", which tells the algorithm to choose the gamma parameter automatically

**Table 17** shows the model's Classification Report and **Figure 30** shows the model's ROC Curve and Confusion Matrix.

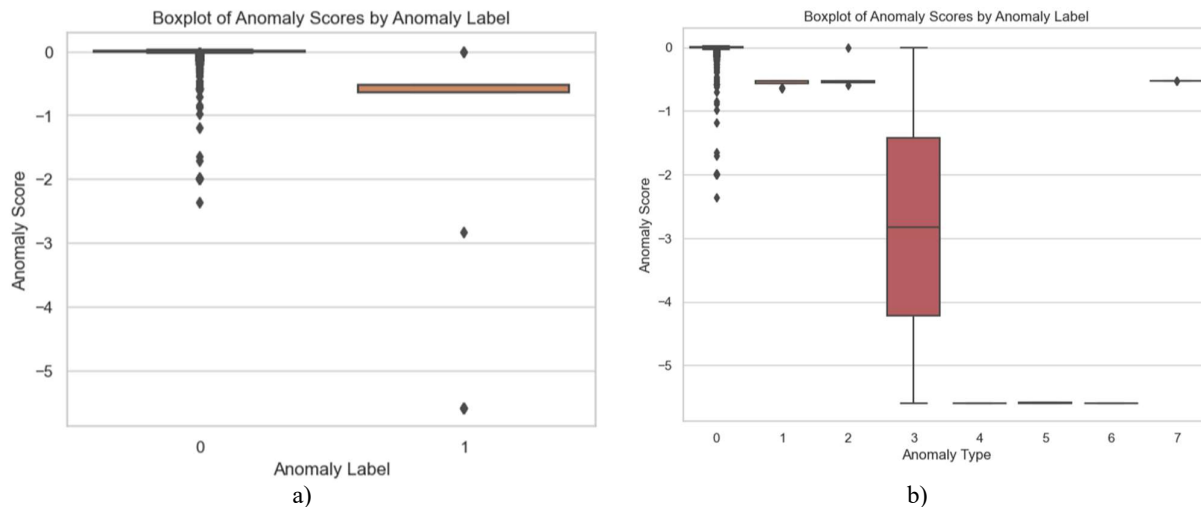
**Table 17:** One-Class SVM Classification Report

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	0.91	0.95	2854
1	0.11	0.94	0.19	34
Accuracy			0.90	2888
Macro Avg	0.55	0.92	0.57	2888
Weighted Avg	0.99	0.91	0.94	2888



**Figure 30:** One-Class SVM ROC Curves and Confusion Matrix

**Figure 31** shows the boxplots of the anomaly scores per anomaly class and per anomaly type. The model worked relatively worse at capturing anomalies of types 1, 2, and 7 than the remaining types.

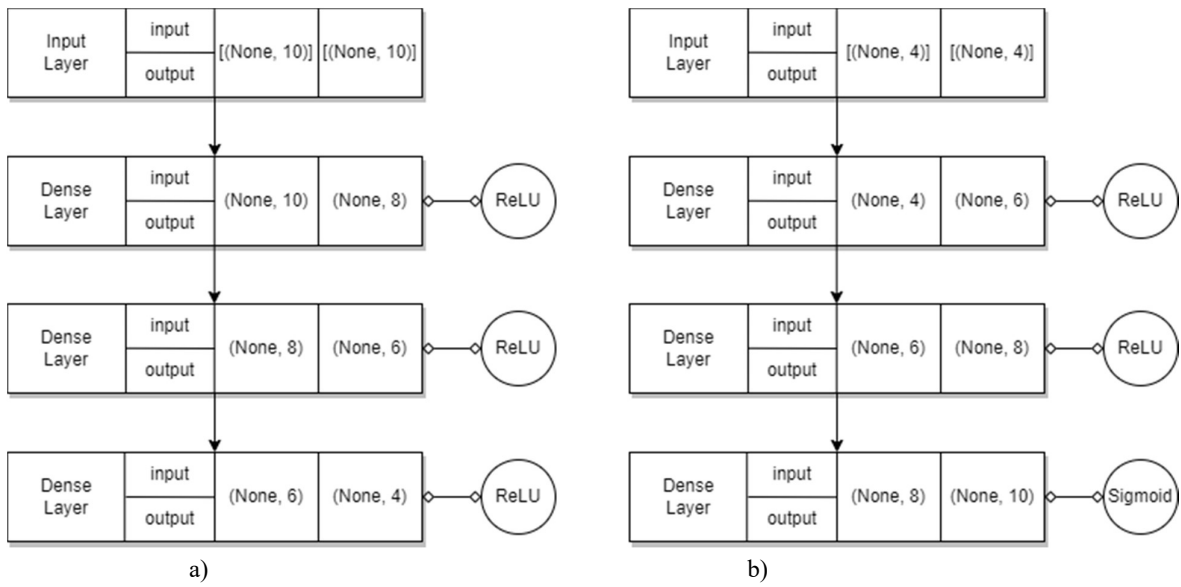


**Figure 31:** One-Class SVM Anomaly Score Boxplots per a) Anomaly Class and b) Anomaly Type

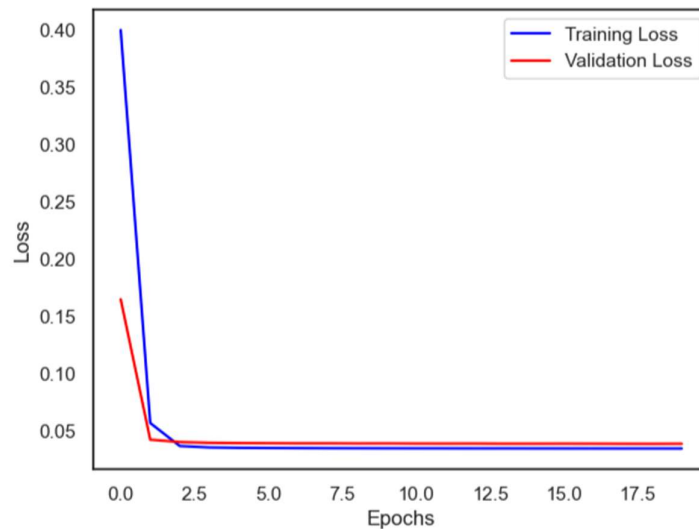


## Auto-Encoder

The Auto-Encoder model consists of two neural networks: the encoder and the decoder. The encoder network has three hidden layers with 8, 6, and 4 units, respectively, and use the *ReLU* as the activation function. The decoder network has three hidden layers, the first and the second layers with 6 and 8 units, and use a *ReLU* activation function, while the last having the same number of units as the input data's dimensions and uses a *Sigmoid* activation function. The autoencoder is trained on 20 epochs. A summary of the Auto-Encoder model is presented in **Figure 32** and the training process is shown in **Figure 33**. We can see that the training and validation losses decrease synchronously with the number of epochs.



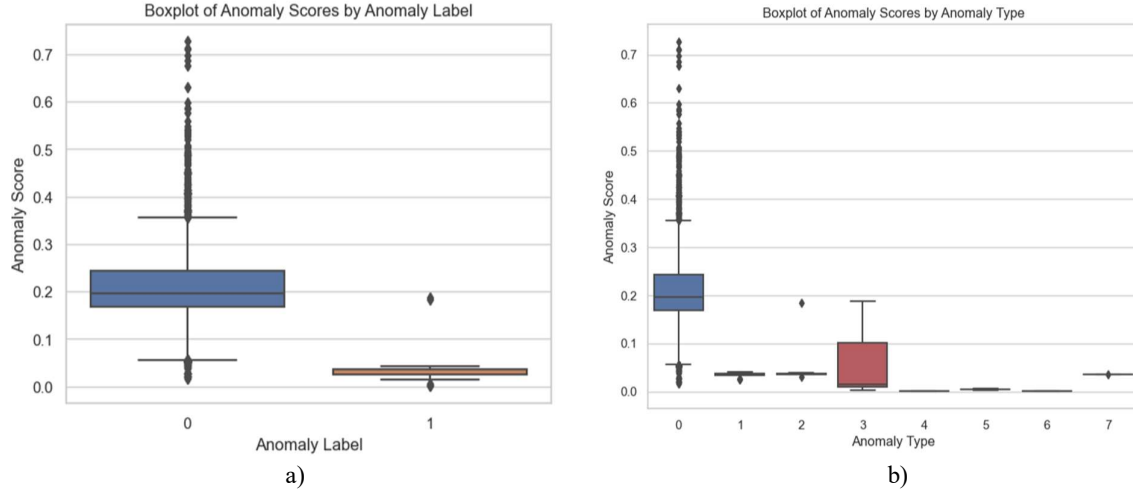
**Figure 32:** Auto-Encoder model architecture a) encoder composition, b) decoder composition



**Figure 33:** Auto-Encoder's Training and Validation Losses for 20 epochs

The Auto-Encoder model was fine-tuned by manually adjusting the loss threshold. The loss threshold is a value that determines how similar the reconstructed data is to the original data. A higher loss threshold will result in more data points being classified as anomalies. Through iterative experimentation, we identified the loss threshold that yielded the best performance in detecting anomalies, which tuned out to be 3%.

**Figure 34** shows the boxplots of the anomaly scores per anomaly class and per anomaly type. The Auto-Encoder performed well at capturing all anomaly types, except for some anomalies of type 3.

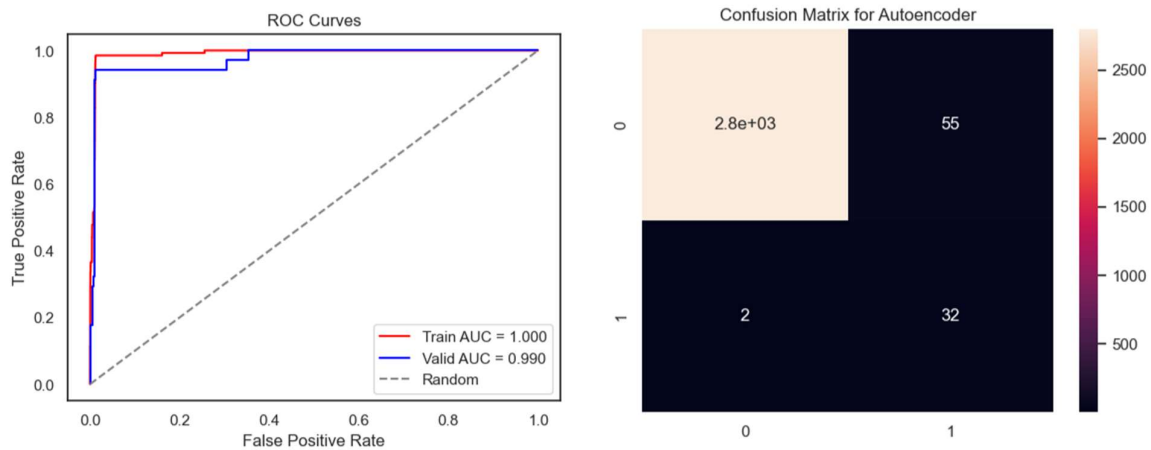


**Figure 34:** Auto-Encoder Anomaly Score Boxplots per **a)** Anomaly Class and **b)** Anomaly Type

**Table 18** shows the model’s Classification Report and **Figure 35** shows the model’s ROC Curve and Confusion Matrix.

**Table 18:** Auto-Encoder Classification Report

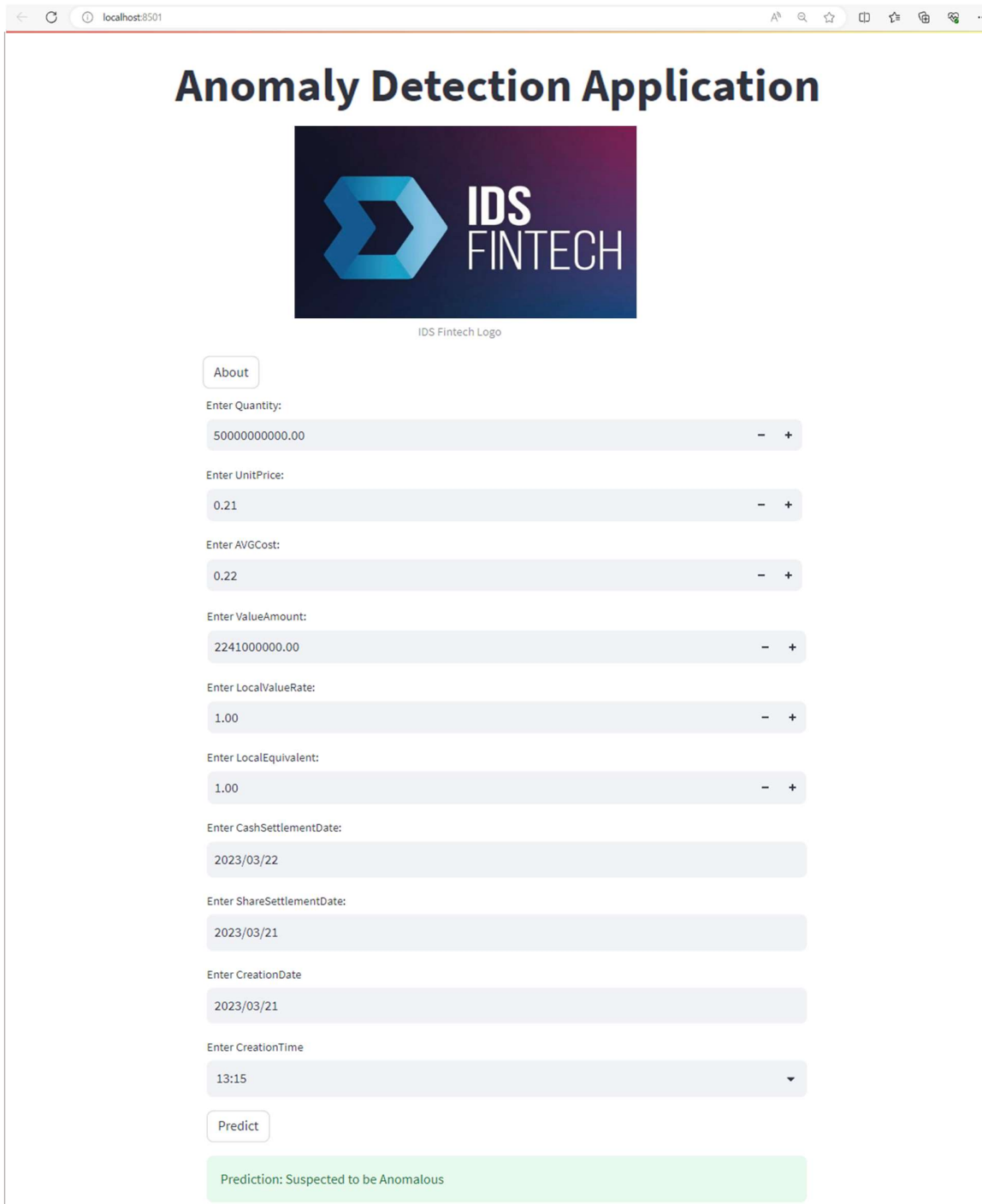
	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
0	1.00	0.98	0.99	2854
1	0.37	0.94	0.53	34
Accuracy			0.98	2888
Macro Avg	0.68	0.96	0.76	2888
Weighted Avg	0.99	0.98	0.98	2888



**Figure 35:** Auto-Encoder ROC Curves and Confusion Matrix

### 4.3 Deployment on Streamlit

A screenshot of the Streamlit web application which shows the deployment of the selected model is displayed in **Figure 36**. The webpage requests from the user to input features of a transactional record, then predicts whether the transaction is anomalous or not based on the chosen model.



The screenshot shows a web browser window with the address bar displaying 'localhost:8501'. The application title is 'Anomaly Detection Application'. Below the title is the 'IDS FINTECH' logo, which consists of a blue stylized 'D' icon and the text 'IDS FINTECH'. Below the logo is the text 'IDS Fintech Logo'. The application features a series of input fields for transactional data, each with a label and a value:

- Enter Quantity: 5000000000.00
- Enter UnitPrice: 0.21
- Enter AVGCost: 0.22
- Enter ValueAmount: 2241000000.00
- Enter LocalValueRate: 1.00
- Enter LocalEquivalent: 1.00
- Enter CashSettlementDate: 2023/03/22
- Enter ShareSettlementDate: 2023/03/21
- Enter CreationDate: 2023/03/21
- Enter CreationTime: 13:15

At the bottom of the input fields is a 'Predict' button. Below the button is a green box containing the text 'Prediction: Suspected to be Anomalous'.

**Figure 36:** Screenshot of the Streamlit Web Application

# 5 Discussion

In this section, we compare the performances of the different models based on the previously selected evaluation metrics. During our comparison, we will take into consideration other criteria which directly affect our choice, such as the model's feasibility and operational efficiency.

## 5.1 Supervised Models

In *Table 19*, we compare the performances of the eight tuned supervised models, listing the counts of their true-negatives (TN), false-negatives (FN), false-positives (FP) and true-positives (TP), in addition to their Macro-averaged Recall and AUC scores. The best-performing supervised model is highlighted in bold.

**Table 19:** Evaluation Comparison of the Supervised Models

No.	Algorithm	TN	FN	FP	TP	Macro-averaged Recall	ROC - AUC
1	Logistic Regression	2850	3	4	31	0.96	0.975
2	K-Nearest Neighbors	2854	4	0	30	0.94	0.985
3	Gaussian Naïve Bayes	2827	3	27	31	0.95	0.976
4	Support Vector Machine	2850	3	4	31	0.96	0.975
5	Decision Tree	2854	2	0	32	0.97	0.985
6	Random Forest	2854	3	0	31	0.96	0.979
7	Extreme Gradient Boosting	2854	1	0	33	0.99	0.979
8	<b>Artificial Neural Networks</b>	<b>2854</b>	<b>1</b>	<b>0</b>	<b>33</b>	<b>0.99</b>	<b>0.99</b>

Among the supervised models, we observe that several achieved very close performances in detecting anomalies. However, the standout performer is the Artificial Neural Networks model, achieving near-perfect accuracy with a Macro-averaged Recall and a ROC – AUC scores of 0.99.

## 5.2 Unsupervised Models

The unsupervised models did not use the label class during the learning process. However, they used the anomaly label class to for evaluation purposes. *Table 20* shows a comparison of the unsupervised models' performances, using also the previously selected evaluation metrics. The best-performing unsupervised model is highlighted in bold.

**Table 20:** Evaluation Comparison of the Unsupervised Models

No.	Algorithm	TN	FN	FP	TP	Macro-averaged Recall	ROC - AUC
1	Isolation Forest	2836	25	18	9	0.63	0.965
2	Local Outlier Factor	2567	16	287	18	0.71	0.655
3	One-Class SVM	2590	2	264	32	0.92	0.976
4	<b>Auto-Encoder</b>	<b>2799</b>	<b>2</b>	<b>55</b>	<b>32</b>	<b>0.96</b>	<b>0.99</b>

The One-Class Support Vector Machine and Auto-Encoder models showcase noteworthy performances. The One-Class SVM excelled with a remarkable ROC - AUC of 0.976 and a Macro-averaged Recall of 0.92. A slightly more impressive is the Auto-Encoder model, attaining ROC - AUC and Macro-averaged Recall scores of 0.99 and 0.96, respectively

### **5.3 Optimal Model for our Specific Objective**

Ultimately, we choose the Auto-Encoder as the optimal model for this anomaly detection task. While the XGBoost and the ANN both resulted in similar evaluation scores, the Auto-Encoder is more appropriate for our specific purpose and domain. Despite being trained without labeled features, the Auto-Encoder model effectively learned the inherent patterns of the data's normal state. Its performance surpassed that of other unsupervised models, successfully identifying 32 out of 34 anomalies in the validation dataset. Additionally, the model flagged 55 instances of suspected anomalies. This information holds significance as the data, which is assumed to be normal, may have contained 1-2% anomalies as stated in the Data Understanding subsection of the Methodology.

The Auto-Encoder model's strength lies in its unsupervised nature, which aligns well with the challenges posed by anomaly detection in financial data. Anomalies are inherently rare occurrences in such datasets, often making up a small fraction of the overall data. This rarity poses a challenge for supervised models, as they may not have enough labelled examples of anomalies to learn from. In contrast, the Auto-Encoder functions by understanding the normal patterns within the data without requiring labelled anomalies. By compressing normal input data into a lower-dimensional space and then reconstructing it, the model effectively learns the underlying structure of normal data.

In the financial domain, anomalies can take various forms, from unusual transaction timings to unexpected currency exchange rates to many different cases which we did not include in our study. The Auto-Encoder's ability to capture subtle deviations and patterns within the data makes it well-suited for this context. Since the model is designed to learn the intrinsic structure of normal data, it is capable of identifying anomalies that do not conform to these learned patterns, regardless of their specific characteristics. After all, we aim to tailor the anomaly detector to the normality of the data, and not the abnormality within it.

Unlike models that might be tailored to specific types of anomalies, the Auto-Encoder is not constrained by fixed concepts of what anomalies should look like, but is constrained by what normal transactions should look like. Therefore, it is capable of identifying anomalies of varying natures and complexities, which is crucial for financial security and operational efficiency.

## 6 Conclusion

In conclusion, this business analytics capstone project addressed the critical need for detecting and preventing transactional anomalies in IDS Fintech's financial data. With the increasing complexity of financial transactions, the potential for errors and anomalies has risen, posing significant risks to financial institutions. The collaborative effort between IDS Fintech and this project aimed to develop a sophisticated anomaly detection model using data analytics and machine learning techniques.

Throughout the project, the methodology involved comprehensive data preparation, feature engineering, and model training. The transactional dataset, collected from IDS Fintech's portfolio management system, was initially assumed to be normal. To enrich the dataset and facilitate a comprehensive evaluation, synthetic anomaly records of seven distinct types were introduced with the invaluable guidance of domain experts. This augmentation not only expanded the scope of models and evaluation methodologies but also enabled a more robust assessment of the anomaly detection techniques. The dataset was analyzed to identify relevant features and patterns and new features were extracted to capture various anomaly types, such as transactions occurring outside working hours, deviations in unit prices, abnormal settlement dates, and more. These features were carefully engineered to improve the model's ability to detect anomalies accurately. The numerical features were scaled and the features with a highest predictive power were selected.

Both supervised and unsupervised models were trained to detect anomalies. Among the supervised models, the Artificial Neural Networks model stood out with near-perfect accuracy. Among the unsupervised models, the Auto-Encoder exhibited exceptional performance by capturing the underlying patterns of normal data and effectively identifying anomalies. Considering the rarity and diverse nature of anomalies in financial data, the Auto-Encoder's ability to identify deviations from normal patterns without being confined to specific anomaly characteristics made it the most suitable choice for this task.

The strength of the Auto-Encoder model lies in its unsupervised nature, aligning seamlessly with the challenges encountered in detecting anomalies within financial data. Given the fact that anomalies are inherently rare occurrences in such datasets, constituting a small portion of the overall data and the fact that labeling all the records is practically impossible due to their fast frequencies, supervised models often struggle with limited labeled examples of anomalies for effective learning. In contrast, the Auto-Encoder operates by grasping the inherent patterns of normal data, eliminating the need for labeled anomalies. By compressing standard input data into a lower-dimensional space and then reconstructing it, the model accurately learns the underlying structure of normative data.

In the financial realm, anomalies manifest in diverse ways, encompassing atypical transaction timings, unexpected currency exchange rates, and various other scenarios that may not have been explicitly covered in our study. The Auto-Encoder's ability to capture subtle deviations and patterns within the data makes it particularly well-suited for this context. As the model is engineered to internalize the intrinsic structure of typical data, it acquires the capacity to spot anomalies that diverge from these learned patterns, irrespective of their specific attributes. The ultimate goal is to customize the anomaly detector to the data's normality rather than the anomalies it incorporates.

The successful implementation of the Auto-Encoder model demonstrates its potential to address the challenges of anomaly detection in financial data. By leveraging this model, IDS Fintech can enhance operational efficiency, minimize financial risks, and reinforce customer trust.

A notable limitation of the selected Auto-Encoder model is its lack of interpretability. While the model effectively identifies anomalies at the row level, it does not explicitly pinpoint the specific feature within the record that contains the anomaly. This limitation could hinder the practicality of implementing corrective actions. Future research could explore the development of explainable anomaly detection methods capable of identifying cell-level anomalies within a record. Methods like De-Noising Auto-Encoders could be investigated to not only detect cell anomalies but also provide estimations of expected values with confidence levels. This advancement would not only enhance the interpretability of the model's findings but also provide suggestions for correcting anomalies within transactions.

By offering actionable insights and guidance on rectifying anomalies, financial institutions like IDS Fintech could further enhance their operational strategies, security and decision-making processes.

## 7 References

- Ahmed, M., Mahmood, A. N., & Islam, M. R. (2016). A survey of anomaly detection techniques in the financial domain. *Future Generation Computer Systems*, 55, 278-288.
- Alghushairy, O., Alsini, R., Soule, T., & Ma, X. (2020). A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing*, 5(1), 1.
- Anandakrishnan, A., Kumar, S., Statnikov, A., Faruque, T., & Xu, D. (2018, January). Anomaly detection in finance: editors' introduction. In *KDD 2017 Workshop on Anomaly Detection in Finance* (pp. 1-7). PMLR.
- Bakumenko, A., & Elragal, A. (2022). Detecting anomalies in financial data using machine learning algorithms. *Systems*, 10(5), 130.
- Becirovic, S., Zunic, E., & Donko, D. (2020, March). A Case Study of Cluster-based and Histogram-based Multivariate Anomaly Detection Approach in General Ledgers. In *2020 19th International Symposium Infoteh-Jahorina (INFOTEH)* (pp. 1-6). IEEE.
- Brownlee, J. (2016). *XGBoost With python: Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery.
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), 1-58.
- Chollet, F., & others. (2015). Keras. *GitHub*. Retrieved from <https://github.com/fchollet/keras>



- Crépey, S., Lehdili, N., Madhar, N., & Thomas, M. (2022). Anomaly Detection in Financial Time Series by Principal Component Analysis and Neural Networks. *Algorithms*, 15(10), 385.
- Jijo, B.T., Abdulazeez, A.M. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. *J. Appl. Sci. Technol. Trends*. 2, 20–28
- Li, G., & Jung, J. J. (2021). Dynamic relationship identification for abnormality detection on financial time series. *Pattern Recognition Letters*, 145, 194-199.
- Liu, F. T., Ting, K. M., & Zhou, Z. H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1), 1-39.
- Liu, J., Yang, D., Zhang, K., Gao, H., & Li, J. (2023). Anomaly and change point detection for time series with concept drift. *World Wide Web*, 1-24.
- Morozov, I. (2016). *Anomaly detection in financial data by using machine learning methods* (Doctoral dissertation, Hochschule für angewandte Wissenschaften Hamburg).
- Nourbakhsh, A., & Bang, G. (2019). A framework for anomaly detection using language modeling, and its applications to finance. *arXiv preprint arXiv:1908.09156*.
- Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2), 1-38.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ..., Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc.

- Retrieved from <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, *JMLR* 12, 2825-2830.
- Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1), 3-14.
- Rish, I. (2021). An Empirical Study of the Naïve Bayes Classifier. In *Proceedings of the IJCAI 2001 Work Empir Methods Artif Intell, Seattle, WA, USA, 4–10*; Volume 3
- Schlör, D. (2022). *Detecting Anomalies in Transaction Data* (Doctoral dissertation, Universität Würzburg).
- Sublime, J., & Kalinicheva, E. (2019). Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the Tohoku tsunami. *Remote Sensing*, 11(9), 1123.
- Sutton, O. (2012). Introduction to k nearest neighbour classification and condensed nearest neighbour data reduction. *University lectures, University of Leicester*, 1.
- Yıldız, K., Dedebeek, S., Okay, F. Y., & Şimşek, M. U. (2022, September). Anomaly Detection in Financial Data using Deep Learning: A Comparative Analysis. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-6). IEEE.
- Zhao, Y., Nasrullah, Z. and Li, Z., 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of machine learning research (JMLR)*, 20(96), 1-7.