Overcoming Imbalanced Class Distribution and Overfitting in Financial Fraud Detection: An Investigation Using A Modified Form of K-Fold Cross Validation Approach to Reach Representativeness

by

Joao Batista Rocha Bezerra Junior
B.Sc., Universidade Federal Fluminense, 2008

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Overcoming Imbalanced Class Distribution and Overfitting in Financial Fraud Detection: An Investigation Using A Modified Form of K-Fold Cross Validation Approach to Reach Representativeness

by

Joao Batista Rocha Bezerra Junior
B.Sc., Universidade Federal Fluminense, 2008

Supervisory Committee

Dr. Daniela Damian, Supervisor
(Department of Computer Science)

Dr. Adam Murray, Supervisor
(Department of Computer Science)

ABSTRACT

According to the Internet Crime Report 2022, the number of complaints and the amount of financial losses from 2018 to 2022 show the total of $27.6 billion dollars, in 3.26 million complaints. Technology has been in development by institutions interested in mitigating cybercrimes, and researchers have been contributing with them to keep ahead the fraudulent systems. Machine learning and deep learning are being applied in a variety of studies to understand and learn how to avoid fraudulent transactions in real-world financial networks from financial institutions, through the use of past transactions. This thesis proposes to use a modified version of k-fold cross-validation technique (Full Sets approach) applied to the PaySim synthetic dataset and submit it to a neural network model, and compare the results to one method of splitting that uses five folds of 20% of the dataset each fold applied to the same model, and then compare it to the machine learning algorithms Random Forest (RF), Logistic Regression (LR), and AdaBoost (AB). The measurements scores applied to evaluate the performances of the models are accuracy, precision, recall, F1 score, specificity, AUC-ROC, and PRC.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Daniela Damian for her guidance and support throughout my Master's degree journey, faith in my abilities, openness to my ideas, precious insights that have significantly influenced my academic and personal growth, and being the responsible for making my accomplishment possible.

Dr. Adam Murray for his support, and relentless hard work throughout the course of my thesis development. His keen insights, meticulous attention to detail, and rigorous standards of scholarship have challenged me to push my boundaries to the highest level ever. Through his detailed guidance, I have gained a deeper understanding of my research subject.

The SEGAL Lab members for their support and friendship throughout my journey.

The University of Victoria for providing an environment that offers academic growth, innovation, and critical thinking, an exceptional faculty that has enriched my learning experience and broadened my perspective, and the administrative staff whose effort creates a thriving academic ecosystem. I am proud to become part of the esteemed alumni from this institution.

My son Joao Guilherme for his love and caring to me, and my daughter-in-law Yvyn Aldren and her mother Silvana for being my son's family with so much love, and still his grandmother Joanita with her dedication and love to my son, making me happy.

Integrity is doing the right thing, even when no one is watching.
C.S. Lewis

Honesty is the first chapter in the book of wisdom.
Thomas Jefferson

DEDICATION

I dedicate this thesis to my beloved son, Joao Guilherme Lino Rocha Bezerra, and my daughter-in-law, Yvyn Aldren. They have been my driving force and inspiration throughout my life. This dedication also stands as a tribute to Edvirges Maria Lino, my son's mother, though no longer with us in person, continues to guide him with her spirit.

Chapter 1

Introduction

The objective of this thesis is to introduce the Full Sets approach applied to the PaySim dataset. The primary focus is to address two significant challenges encountered in financial fraud detection: overfitting [1], due to when a model does not generalize, and subsets representing the dataset with different class proportion distributions. The proposed technique seeks to handle these issues by ensuring that class distribution is proportional across subsets, thereby improving representativeness and preventing overfitting.

Unbalanced class distribution in financial fraud detection datasets occurs when the number of fraudulent transactions is much smaller than the number of non-fraudulent transactions, which happens in most of the cases and it cannot be changed since it is the nature of this kind of dataset with financial frauds against legitimate transactions. However, this unbalance can lead to biased models that fail to detect fraudulent activities, therefore other methods should be applied to compensate. Overfitting, on the other hand, occurs when a model is too complex and fits the training data too closely, resulting in poor performance on generalization.

To address these challenges, this thesis proposes using a simple neural network [2] as the model. This research investigates the utilization of representativeness techniques. One such technique is a type of stratified sampling, which involves the proposed modified form of k-fold cross-validation applied to the dataset. The purpose of this technique is to ensure that the proportion of observations in each class or category remains roughly the same across all subsets. By doing so, it guarantees that the resulting subsets are representative of the overall dataset.

This thesis applies a range of traditional machine learning [2] techniques, including Random Forest (RF) [1] and Logistic Regression (LR) [3], against a simple neural

network with three layers [4], to investigate the effectiveness of these approaches and perform comparative analysis between these models and also confronting the results of a prior study that applied the convolutional neural network on the same dataset.

1.1 Motivation

Financial frauds can have significant impacts on individuals, businesses, and society as a whole. Victims of fraud can suffer financial losses, damage to their credit scores, and emotional distress. In addition, fraud can decrease trust in financial institutions and systems, which can have broader implications for the economy and society.

According to a report by the Association of Certified Fraud Examiners (ACFE) [5], the global cost of occupational fraud alone is estimated to be around $3.7 trillion annually. This includes fraudulent activities such as asset misappropriation, corruption, and financial statement fraud. The report highlights the importance of taking proactive measures to prevent and detect fraud, including implementing strong internal controls and conducting regular fraud risk assessments.

Furthermore, financial fraud has become increasingly sophisticated with the rise of digital technologies and the Internet. Cybercriminals can use a range of techniques, such as phishing, malware, and social engineering, to gain access to sensitive information and steal money. As such, it is crucial for individuals and organizations to stay vigilant and take steps to protect themselves from online fraud.

The Internet Crime Complaint Center (IC3) [6], which is a partnership between the Federal Bureau of Investigation (FBI) and the National White Collar Crime Center (NW3C), serves as a central point of contact for individuals to report suspected cybercrimes to the FBI. On the website, individuals can submit complaints related to various types of Internet crimes, such as identity theft, hacking, online scams, phishing, and child exploitation. The IC3 analyzes the complaints and provides information to law enforcement agencies to help them investigate and prosecute cybercriminals. The portal also provides tips and resources for Internet safety, including ways to protect yourself from online scams and fraud, as well as information on how to report cybercrime and get help if you are a victim.

The Figure 1.1 below presents a graph from the Internet Crime Report 2022 [7] with the statistics about fraud complaints and financial losses dated from 2018 to 2022, showing the amount of $27.6 billion dollars, in 3.26 million complaints. Over this 5-year period, the IC3 has received an average of 652,000 complaints per year. These complaints address a large amount of Internet scams affecting victims worldwide.

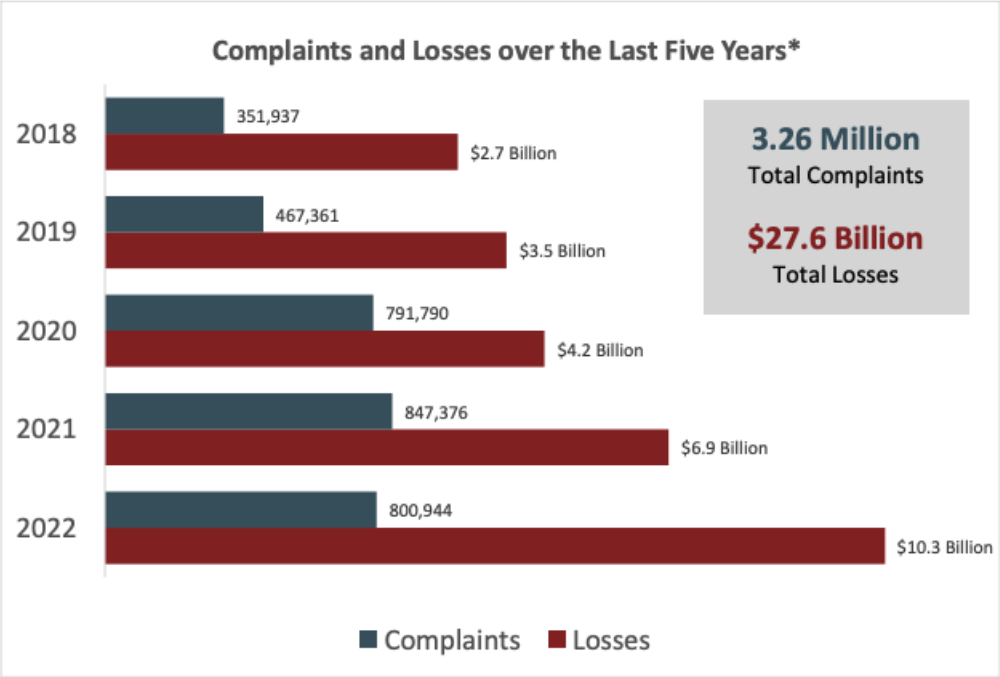

Figure 1.1: Chart showing yearly and aggregate data for complaints and losses over the years 2018 to 2022.

According to the PwC's Global Economic Crime and Fraud Survey 2022 [8], financial frauds have a significant impact on society. They not only result in significant financial losses for individuals and businesses, but also decrease trust in financial institutions and systems. In addition, they can have larger societal consequences, such as undermining economic growth, reducing tax revenues, and diverting resources away from productive activities. Financial frauds can take many forms, including identity theft, Ponzi schemes, insider trading, and accounting fraud.

To reduce financial losses caused by fraud, researchers have conducted extensive studies on various aspects of fraud, including its causes, detection, prevention, and punishment. For example, some studies have focused on the psychological and social

factors that lead individuals to engage in fraud, such as greed, pressure to meet performance targets, and lack of moral and ethical values. Other studies have examined the effectiveness of various fraud detection and prevention measures, such as data analytics, employee training, and regulatory enforcement.

The research on financial fraud has led to a better understanding of the nature and extent of the problem, as well as the most effective ways to mitigate its impact. By applying the insights and recommendations from these studies, individuals, businesses, and governments can reduce their vulnerability to financial fraud and improve the integrity of financial systems. However, fraud remains an ongoing challenge, and continued research and innovation are necessary to stay ahead of the evolving tactics and strategies of fraudsters.

Cybersecurity Ventures [9] is a cybersecurity research and market intelligence firm that provides cybersecurity statistics, insights, and reports on cybersecurity trends, threats, and opportunities. It publishes a range of reports and publications on cybersecurity, including the Cybersecurity 500, a ranking of the world's hottest and most innovative cybersecurity companies, and the Cybercrime Magazine, an online publication that covers news and trends in cybercrime and cybersecurity.

In 2022, Cybersecurity Ventures published the *2022 Cybersecurity Almanac* [10], and here are some insights inside this report related to cybercrime damage:

- Digital ad fraud is rising sharply. The ad industry loses approximately $51 million per day due to ad fraud and by 2023 that number will skyrocket to $100 billion annually, according to an estimate featured in Bloomberg Law.

- Cybercrimes are vastly undercounted because they aren't reported — due to embarrassment, fear of reputational harm, and the notion that law enforcement can't help (amongst other reasons). Some estimates suggest as few as 10 percent of the total number of cybercrimes committed each year are actually reported.

- Organized cybercrime entities are joining forces, and their likelihood of detection and prosecution is estimated to be as low as 0.05% in the U.S., according to the World Economic Forum's 2020 Global Risk Report [11].

1.2 Research Questions (RQ)

This thesis aims to address the challenges of overfitting and inadequate representation of subsets in financial fraud detection by proposing a modified version of the k-fold cross-validation method applicable to the preprocessed PaySim dataset [12], besides guaranteeing that all data is applied in the training, validation and test phases, not in the same round but after all the rounds of subsets are applied one by one to the model. The goal of the proposed technique is to improve representativeness and prevent overfitting by ensuring proportional class distribution across subsets, when using a neural network as the model performing the classification. As the measurement of results, the traditional measurements are used to state the performance of the model. In the comparison to other models, a reduced splitting that uses 20% of the dataset for each subset is used. Additionally, the Full Sets approach is compared to three machine learning approaches and also to prior research that used Convolutional Neural Network on the same PaySim dataset.

As a guide for this thesis, there are three research questions:

- **RQ1:** Which approach has the best results, the Iterations method that use 5 sets of 20% of dataset or the Full Sets approach that use sets of 100% of dataset applied to the same neural network model?
- **RQ2:** Which traditional machine learning method performs better on this specific dataset, Random Forest, Logistic Regression or AdaBoost?
- **RQ3:** How does the best performing approach submitted to the neural network model compare to the superior performing approach from Random Forest, Logistic Regression or AdaBoost?

There are two hypotheses for this thesis:

- **HP1:** The approach that has the best performance is the Full Sets, over the Iterations and the machine learning techniques.

  - Justification: The technique applied in the Full Sets approach for the dataset format differ from the one applied in the Iterations method as follows: In the Full Sets approach, each of the five sets in the dataset contains all the data, with the classes proportionally distributed among the training, validation, and testing sets. Once all five Full Sets have been applied, the entire dataset has been used for training, validation, and testing, thus increasing the amount of data available for training. In contrast, the Iteration approach only uses 20% of the dataset for each set.

- **HP2:** Random Forest has the best performance comparing to Logistic Regression and AdaBoost.

  - Justification: Random Forest is a powerful machine learning method due to its ability to handle noise, missing data, and feature selection, as well as its ability to combine multiple decision trees for improved accuracy: Each decision tree is created using a random subset of the data and features, so the trees are diverse. By combining the output of many diverse trees, the final prediction is less likely to be affected by noise in the data. Research by Saheed [13] used Random Forest approach.

By the end of this thesis, the three research questions will be answered and the two hypothesis will be confirmed or denied.

1.3 Research Contributions

This thesis applies the technique k-fold cross validation with a slight difference in the data splitting which is done manually before applying the model.

The contributions of this thesis, lie not only in the comparative analysis of machine learning approaches, but also in raising crucial awareness regarding the limitations and potential issues with synthetic datasets. Despite the frequent use of such datasets in research, this study revealed potential issues with their representation of real-world scenarios. As shown, models trained on these datasets may yield impressive testing results but fall short in practical applications due to lack of real-world representation. This discrepancy underscores the importance of rigorous dataset scrutiny for ensuring reliable model performance and generalization. Furthermore, the divergence between

expected and actual performance observed in this study calls into question the methodology behind the synthetic data generation. This prompts further investigation into these techniques, cultivating a more critical approach to dataset selection within the field.

An additional contribution from this thesis is that it opens up an avenue for future research, providing a pathway foundation for the continuation of this thesis, aiming the evaluation of the synthetic PaySim dataset.

1.4 Thesis Overview

This thesis is organized as follows:

Chapter 1: Introduction presents the context of the research problem, and the motivation for this thesis, introduces the research questions, discusses the contributions and shows the thesis overview.

Chapter 2: Literature Review provides a background of the concepts and techniques applied in this thesis, and presents the related work.

Chapter 3: Methodology describes the dataset and the preprocessing steps, and presents the new approach of splitting the data, and the models used in the classification processes.

Chapter 4: Results Analysis provides all the results of the methods used in the classification phase, measurements analysis and graph analysis, including the tables with the respective performances.

Chapter 5: Discussions presents the results summary, findings, threats to validity, prior research comparison, and future work.

Chapter 6: Conclusion contains a recapitulation of my research as a summary, the main findings, implications, reflection and a conclusion of this thesis.

Chapter 2

Literature Review

2.1 Introduction

This is the literature review of a thesis that aims to use machine learning and deep learning [1] approaches to detect fraudulent transactions based on previous historical data. This chapter provides background information on various concepts and techniques related to machine learning in financial fraud detection.

Machine learning has started to be systematically used for detecting financial fraud by several researchers decades ago, and the approaches and techniques have evolved with the advancement of technology over time.

Green, Brian Patrick and Choi [14], in 1997, developed an effective neural network fraud classification model employing endogenous financial data, and during the training of the model, a neural network learns the pattern of input data for a fraud and nonfraud sample. These results from the study supported future use of neural network as a fraud-risk assessment tool.

He, Hongxing, and Graco [15] applied the K-nearest neighbor algorithm in 1999 to a medical fraud detection problem. In classification tasks, this algorithm compares a new data point with the K-nearest data points (as determined by the user) in the training set. The class label of the new data point is then assigned based on the majority class label among the K-nearest neighbors. They combined this algorithm with a genetic algorithm to solve the medical fraud detection problem.

Prodromidis [16], also in 1999, describes an AI-based approach that combines inductive learning algorithms and meta-learning methods as a means to compute

accurate classification models for detecting electronic fraud, and through experiments performed on actual credit card transaction data supplied by two different financial institutions, the approach is evaluated and its utility demonstrated.

Machine learning algorithms such as neural networks, support vector machines, and random forests have been developed and applied to financial fraud detection for decades. Deep learning techniques such as convolutional neural networks and recurrent neural networks have also been employed in financial fraud detection. These techniques have shown promising results in identifying complex patterns in financial data that may be indicative of fraudulent activity.

The use of machine learning in financial fraud detection has become increasingly prevalent due to the exponential growth of financial data and the need for more advanced techniques to identify fraudsters who are becoming increasingly sophisticated in their techniques.

Historical data is crucial in financial fraud detection because it provides a baseline of normal behavior and transactions. By analyzing past transactions, patterns, and trends, anomalies can be identified indicating a fraudulent activity. This thesis wants to address some aspects involved in this process of using machine learning and deep learning approaches to detect fraudulent transactions, applied to the PaySim dataset, which is a simulation of financial transactions based on real-world data and validated.

The primary objective of this thesis is to concentrate on data preparation, starting from the raw dataset and progressing through the preprocessing methods utilized by Johari [17] and Mubalaike [18]. It subsequently involves the application of a modified k-fold cross-validation [1] data distribution technique for data splitting, which results in k=5. This means the data is partitioned into five distinct folds or sets (each set or fold represents 100% of the total dataset), the specifics of which are elaborated in Chapter 3. The process culminates with the employment of a simple three-layered neural network model, followed by performance measurement. This method is called Full Sets approach. A second data formatting is applied to the preprocessed dataset, which is also described in Chapter 3, that uses a full dataset split in 5 subsets with 20% of the full dataset, called Iterations approach. It differs from the Full Sets approach in that each of its subsets comprises only one-fifth of the full dataset. Its class distribution is proportionally maintained across the five sets to ensure representativeness. The idea is to assess the Full Sets approach, initially by comparing it with the Iterations approach. The goal of this comparison is to evaluate if the Iterations

approach could be used in the classification successfully, in case it has better performance in correctly predicting fradulent and legitimate transactions, than the Full Sets method. It would be an advantage since each subset has 1/5 of the data comparing to the the subset of the Full Sets, meaning less computer power, less time, less data, with the same or better results. In both procedures, a neural network is used to classify each of the five sets separately and then an average number is calculated. In the end, we have average scores for each of the approaches and the comparison of the performance is executed comparing the averages.

A second evaluation is to submit the full dataset, after the preprocessing, to three machine learning methods, Random Forest, Logistic Regression and AdaBoost classifiers, measure the scores of the three models, and compare the best results from the three approaches in this particular dataset to the Full Sets approach.

In the methodology of this thesis, the decision for not utilizing measures such as standard deviation and median was deliberately made, reflecting my choice to adhere to the foundational principles of k-fold cross-validation. K-fold cross-validation traditionally focuses on folds and averages, rather than standard deviation or median. This approach enables a straightforward comparison across multiple iterations of the model. Additionally, one of the objectives is to compare with the methods used by a prior study. Therefore, maintaining consistency in my evaluation methods with previous research is critical. By doing so, I can create a meaningful comparative analysis of the model's performance. This consistency, in turn, provides clearer insights into the distinct contributions and advancements of my study over existing research.

This research is addressing overfitting and lack of representativeness within the dataset subsets. This is detailed in the sequence, the steps related to the dataset preparation of the Full Sets approach, after the preprocessing:

1. Split the dataset in three sets, training, validation and test sets: The dataset is split two subsets, frauds and nonfrauds, and each subset is split in ten parts (10% of the dataset), so we have ten parts of frauds and ten parts (10% of the dataset) of nonfrauds. The training set receives six parts of frauds subsets and six parts of nonfrauds subsets (60% of the dataset), and the four other parts of the fraud and nonfrauds subsets go to the "validation and test sets", two parts of frauds and two parts of nonfrauds for each (20%). Hyperparameters [2] are the settings of the neural network that are not learned during training

but are set before training begins. Examples of hyperparameters include the learning rate, the number of layers in the network, and the number of neurons in each layer. The necessity of using a third set called the validation set is to tune the hyperparameters of the neural network model. The purpose of the validation set is to evaluate the performance of the model during training and help improve the values for the hyperparameters of the model.

2. Each set in the Full Sets approach, is split in 20% test set, 20% validation set, and 60% training set (this format of splitting, 60%/20%/20%, is explained in detail in the Methodology, Chapter 3). Since there are five sets, each one is distributed in such a way that all the data will be part of the training, validation and test sets, after applying the model in all the five rounds, each one for a full set, this will:

    - This is particularly important when the dataset does not have sufficient data for training, as it maximizes the use of available data for both training and testing.
    - Provide a more reliable estimate of performance: This technique (k-fold cross-validation) helps to reduce the impact of random variability in the data and provides a more stable estimate of performance.
    - Help in hyperparameter tuning: The validation set can be used to tune hyperparameters of the model by evaluating the model's performance on different hyperparameter configurations on different folds. This helps to improve the hyperparameters.

3. Instances of both fraudulent and legitimate nature are proportionally distributed among the training, validation, and test sets within each of the five full sets, ensuring that they accurately represent the entirety of the dataset. Implementing this approach enhances the model's capability of detecting variability of the data while maintaining high fidelity to the original data.

The next sections of this Chapter are *Background* containing concepts related to this research and techniques applied in this thesis, and *Related Work* with the research that has are been conducted during the last two decades in the field of machine learning and deep learning used in financial fraud detection.

2.2 Background

2.2.1 Machine Learning in Financial Fraud Detection

Machine learning has been used in the field of financial fraud detection for decades. Financial institutions face a significant challenge in detecting and preventing fraud, as fraudulent activity can be complex and difficult to identify. Machine learning algorithms can help financial institutions to analyze vast amounts of transactional data and identify suspicious patterns of behavior.

One common approach to using machine learning in financial fraud detection is through anomaly detection. Anomaly detection involves identifying normal patterns in data that fit the expected behavior of a typical transaction. Machine learning algorithms can learn from historical data and identify patterns that deviate from the norm, which can indicate potential fraudulent activity. Other approaches to using machine learning in financial fraud detection include supervised learning, where the algorithm is trained on labeled data, and unsupervised learning, where the algorithm identifies patterns in unlabeled data [19].

2.2.2 Supervised Learning

Supervised learning is a type of machine learning in which an algorithm is trained on a labeled dataset, meaning that each data point is associated with a known target or output variable. The goal of supervised learning is to learn a function that can accurately predict the output variable for new, unseen input data. The algorithm uses the labeled data to learn the relationship between the input variables and the output variable and can then generalize to make predictions on new, unseen data.

Supervised learning can be used for a variety of applications, such as image classification, speech recognition, and fraud detection. Some common supervised learning algorithms include linear regression, logistic regression, decision trees, and neural networks. The success of supervised learning models depends on the quality and quantity

of the labeled data used for training. With a sufficient amount of high-quality labeled data, supervised learning algorithms can produce accurate and reliable predictions, making them a powerful tool in many real-world applications [2, 20].

2.2.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) [21], like many other deep learning models, are often referred to as *black box models*. This terminology comes from the complexity of these models and the difficulty of interpreting their inner workings and decision-making processes.

Here are a few reasons why CNNs are considered black box methods:

- Layer Complexity: A typical CNN consists of multiple layers, each performing various transformations on the input data. While we can understand the function of each layer in a broad sense (for instance, convolutional layers apply filters to the input, pooling layers reduce dimensionality, etc.), understanding how these operations collectively contribute to the final prediction is non-trivial.
- Weight Opacity: CNNs involve a large number of weights (parameters) which are adjusted during training. It's challenging to interpret the meaning of each weight in relation to the final output.
- Non-linearity: CNNs, like many deep learning models, are highly nonlinear. This means that small changes in input can sometimes lead to large changes in output, and vice versa. This non-linear relationship makes the models harder to interpret.
- Lack of Rules: Unlike some traditional machine learning models, CNNs don't provide a clear set of rules or decision trees that can be easily interpreted. Instead, they learn complex representations in the data that are hard to express in a human understandable format.

2.2.4 Unbalanced Datasets

An unbalanced dataset is a type of dataset in which the distribution of the target variable or class labels is not evenly distributed. In other words, one class may have

significantly more samples than the other class(es), resulting in a highly unbalanced in dataset. This can be a common occurrence in many real-world scenarios, such as fraud detection, medical diagnosis, or rare event prediction.

For example, consider a binary classification problem where the goal is to predict whether a credit card transaction is fraudulent or not. If only 1% of the transactions are fraudulent, and the remaining 99% are legitimate, then the dataset is unbalanced. This is because the positive class (fraudulent transactions) is significantly underrepresented compared to the negative class (legitimate transactions).

Unbalanced datasets can present significant challenges for machine learning algorithms since they are typically designed to optimize accuracy, which may not be the best metric to evaluate model performance in unbalanced datasets. For example, a model that always predicts the majority class (i.e., the negative class in the credit card fraud example) may achieve high accuracy, but it would fail to identify the minority class (i.e., the positive class), which is often of higher interest.

It is important to note that while balancing the dataset can help improve the performance of the model, it is not always necessary or appropriate. In some cases, the class unbalance may reflect the real-world distribution of the data, and in such cases, it may be more important to focus on optimizing other measurements scores such as precision, recall, or F1-score, which provide a better representation of the model's performance on the minority class [2].

2.2.5 Anomaly Detection

Anomaly detection is an important process in data analysis and machine learning that involves identifying rare or unusual events or observations within a dataset. These anomalies can be indicative of errors, fraud, or other important events that require attention. There are a variety of methods for performing anomaly detection, including statistical methods, machine learning approaches, and deep learning techniques. Anomaly detection is used in a range of applications, including intrusion detection, fraud detection, and medical diagnosis [22].

2.2.6 Real-World Financial Transactions Datasets

Real-world financial transactions datasets contain information on actual financial transactions between individuals, organizations, and governments. These datasets can be used for various purposes, including fraud detection, risk management, and financial forecasting. Here are four examples of real-world financial transactions datasets:

1. PayPal transaction dataset: PayPal is a popular online payment platform that enables people to make and receive payments over the internet. The PayPal transaction dataset contains information on actual transactions made through the platform, including the amount, date, and location of the transaction. This dataset is available for research purposes from PayPal [23].

2. Bitcoin transaction dataset: Bitcoin is a digital currency that is exchanged over a decentralized network. The Bitcoin transaction dataset contains information on actual transactions made on the Bitcoin network, including the amount, date, and location of the transaction. This dataset is available for research purposes from various sources, including the Bitcoin blockchain [24].

3. New York City taxi trip dataset: The New York City taxi trip dataset contains information on actual taxi trips taken in New York City, including the pickup and drop-off locations, the fare, and the tip amount. This dataset is publicly available from the New York City Taxi and Limousine Commission [25].

4. Credit card fraud detection dataset: The credit card fraud detection dataset contains information on actual credit card transactions, including the amount, date, and location of the transaction, as well as other features that can help predict fraudulent transactions. This dataset is available for research purposes from various sources, including Kaggle [26].

2.2.7 Data Preprocessing

Data preprocessing is an important step in machine learning that involves cleaning, transforming, and organizing data before it is used to train a model. The goal is to ensure that the data is in a format that is suitable for analysis and that any errors or inconsistencies are corrected. Some common issues related to data before preprocessing are:

- Incomplete data: Raw data can be incomplete, with missing values, which can cause problems when trying to analyze the data.
- Inconsistent data: Different formats or types of data in different columns or rows, which can make it difficult to analyze.
- Noisy data: Irrelevant data or data with errors that can make it harder to extract useful information, making it difficult for the model to identify meaningful patterns or relationships.
- Outliers: Data points that are significantly different from the other data points in the dataset. Outliers can affect the analysis biasing the algorithm to present results not accurate.
- Duplicate data: Data that may skew[1] the results of the analysis.

Techniques applied to mitigate the issues mentioned previously in the preprocessing of the dataset include data cleaning, data transformation, and data normalization. See below:

- Data cleaning: Removing or correcting any errors or inconsistencies in the data, such as missing values or outliers. Techniques for data cleaning include imputation, which involves filling in missing values, and outlier detection, which involves identifying and removing data points that are significantly different from the rest of the data.
- Data transformation: Converting the data into a format that is suitable for analysis, such as scaling or encoding categorical variables. Techniques for data

[1]Skewness refers to the degree of asymmetry observed in a probability distribution

transformation include feature scaling, which involves standardizing the range of numerical features, and one-hot encoding, which involves converting categorical features into binary features.

- Data normalization: Ensuring that the data is consistent in terms of format and scale, such as converting all data to a common scale or ensuring that data is represented using the same units of measurement.

The choice of preprocessing techniques depends on the specific characteristics of the dataset and the goals of the analysis. A well-preprocessed dataset can significantly improve the accuracy and reliability of machine learning models, while also reducing the risk of errors and bias [27].

Feature extraction is a process of deriving new features (columns) from existing ones in a dataset. The goal is to create a set of more informative and discriminative features that can improve the performance of machine learning models. There are several techniques for feature extraction, including dimensionality reduction, text mining, and image processing.

Dimensionality reduction techniques involve reducing the number of features in a dataset while preserving as much of the original information as possible. Examples of dimensionality reduction techniques include principal component analysis (PCA), linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE).

Text mining techniques involve extracting features from text data, such as the frequency of specific words or the presence of certain patterns. Examples of text mining techniques include bag-of-words models, n-gram models, and topic modeling.

Image processing techniques involve extracting features from image data, such as edge detection, texture analysis, and color histograms. Examples of image processing techniques include convolutional neural networks (CNNs), scale-invariant feature transform (SIFT), and local binary patterns (LBP) [28].

Feature selection is a process of selecting a subset of the original features in a dataset that are most relevant to the prediction task. The goal is to improve the performance of machine learning models by reducing the dimensionality of the data and removing irrelevant or redundant features. There are several techniques for feature selection, including filter methods, wrapper methods, and embedded methods.

Filter methods involve ranking features based on some statistical measure of relevance, such as correlation or mutual information, and selecting the top-ranked features. Wrapper methods involve evaluating different subsets of features using a particular machine learning algorithm and selecting the subset that results in the best performance. Embedded methods involve integrating feature selection into the machine learning algorithm itself, so that features are selected during the model training process [28].

One hot encoding is a technique used in machine learning to represent categorical data as numerical data. Each unique category is converted into a binary vector where each element of the vector represents a category. The vector has a length equal to the number of unique categories, and each element is set to 0 except for the element corresponding to the category, which is set to 1.

For example, if we have a categorical variable "Class" with possible values "fraud", and "nonfraud", we can use one hot encoding to represent the data as follows:

- "fraud": [1, 0]
- "nonfraud": [0, 1]

This encoding allows the data to be used in machine learning algorithms that require numerical input, such as neural networks or decision trees. It also preserves the notion of categorical data in the encoded vectors, making it easy to interpret the results of the algorithm [29].

Feature scaling is the process of transforming the features in a dataset to a common scale to improve the performance of machine learning algorithms. This technique helps to ensure that the features are treated equally during training and prevents features with large values from dominating the learning process.

There are several methods for feature scaling, however the one used in this thesis is below:

- Min-Max Scaling: This method scales the features to a range between 0 and 1. It is calculated by subtracting the minimum value of the feature and dividing by the range of the feature.

The choice of feature scaling method depends on the specific problem and the characteristics of the data. In general, feature scaling is an important step in machine learning that can improve the performance of the algorithm by ensuring that the features are on a common scale. [2, 30].

2.2.8 Training, Validation, and Test Sets

Splitting a dataset into training, validation, and test sets is a crucial step in the process of building and evaluating machine learning models. Here are some reasons why this is important:

- Preventing overfitting: By evaluating the model on a separate validation set, we can tune hyperparameters and prevent overfitting to the training data. This helps to ensure that the model generalizes well to unseen data.
- Assessing model performance: By evaluating the final model on a separate test set, we can get an unbiased estimate of its performance on unseen data. This helps to avoid the issue of "optimistic bias" that can occur when using the same data to both train and evaluate the model.
- Reproducibility: By using a fixed random seed to split the data into subsets, we can ensure that the same subsets are used for each experiment. This makes the experiments more reproducible and allows for fair comparisons between different models and hyperparameters.

The splitting strategy in this thesis involves randomly dividing the dataset into three subsets, with a typical split ratio of 60-20-20 for the training, validation, and test sets, respectively. It is important to ensure that the subsets are representative of the overall dataset and that there is no overlap between them [31, 2].

2.3 Related Work

Zareapoor [32] in 2012 presents survey of various techniques used in credit card fraud detection like Neural Network (NN), Support Vector Machines (SVM), Artificial Immune System (AIS), Decision Tree (DT), K-Nearest Neighbor, and genetic algorithms, and evaluates each methodology based on certain design criteria.

Sundarkumar [33], in 2015, proposes the use of 10-fold cross validation method with multiple traditional machine learning approaches, on a credit card Churn prediction dataset and the decision tree method achieved the highest accuracy, 91.2%.

Anis, Maira and Ali [34], in 2015, works with a credit card dataset and applies the Random Under Sampling (RUS) approach (it works by randomly eliminating instances from the majority class to prevent it from dominating the minority class), along with feature selection, and compared to a previous study that used Decision Tree classifier, and achieved better results using the RUS method.

In 2016, Lopez [35] uses PaySim [12] which is a financial simulator that simulates mobile money transactions based on an original dataset. He presents a solution to simulate mobile money transactions in such a way that they become similar to the original dataset. The technology used is Agent-Based simulation techniques and mathematical statistics, and in the end he shows in this paper that the simulated data can be as promising as the original dataset for research.

In 2017, Alvarez-Jare [36] applied random forest with SMOTE transformation, and had good results with this combination, obtaining 96.15% of true negative results and 94,98% of true positive results. This study has been developed from the investigation of a real Spanish money laundering case in which this expert team have been collaborating, and is the first step to use machine learning to detect financial crime in Spanish judicial process cases.

Zhang, Zhaohui and Zhou [37] in 2018, in proposes a convolutional neural network as a fraud detection model applied on online transactions, and the whole network

consists of a feature sequencing layer, four convolutional layers, pooling layers, and a fully connected layer. Verifying with online transaction data from a commercial bank, and the experimental results are precision at around 91% and recall at around 94%, which increased by 26% and 2%, respectively, comparing with the existing CNN for fraud detection.

In 2018, Mubalaike [18] extracts a Dataset from one month of genuine financial logs of a mobile money service company in Africa, containing about more than six million transactions. The author applies machine learning techniques as ensemble of decision tree (EDT), and deep learning techniques as stacked auto-encoders (SAE) and Restricted Boltzmann Machines (RBM) on the preprocessed data. The performance of generated classifier models is evaluated based on accuracy, sensitivity, specificity, precision, confusion matrix and ROC values, and the results of optimal accuracy are 90.49%, 80.52% and 91.53% respectively, concluding that, based on the results, the restricted Boltzmann machine performs superior than the other techniques.

In 2020, Sekar, Rengarajan, and Manivannan [38] use several machine learning and deep learning approaches to detect credit card fraud. They highlight the importance of fraud detection in financial transactions, and present analysis of the different techniques that can be used for fraud detection. They describe how machine learning techniques such as logistic regression, decision trees, and random forests can be used for credit card fraud detection, and how deep learning techniques such as neural networks and convolutional neural networks can be used for the same purpose. They also compare the performance of these techniques and provide insights into which techniques are best suited for credit card fraud detection.

Misra, Sumit and Thakur [39], in 2020, proposes a two stage model, autoencoder is used to transform the transaction attributes to a feature vector of lower dimension, and this feature vector is used as input of a classifier in the second stage. The experiment is done on a benchmarked dataset, and F1-measure is the metric, and this approach performs better than relying on only classifier.

In 2020, Alghofaili [40] proposes a deep learning-based method for the detection of financial fraud based on the Long Short-Term Memory (LSTM) technique. A real dataset of credit card frauds is used to to evaluate the proposed model, and the results

are compared with an existing deep learning model named Auto-encoder and some other machine learning techniques, and the results indicated a high performance of LSTM where it achieved 99.95% of accuracy.

In 2021, Liu, Xiao and Yan [41], propose a deep learning model called CCFD-Net for credit card fraud detection using 1D-Conv technique and Res-net framework. The dataset used is Vesta's e-commerce transactions [42], and the proposed model performs better than the other ones and it was evaluated in terms of AUC-ROC.

In 2021, Chen [43] proposes the Deep Convolution Neural Network approach to detect financial fraud detection using applying it in a large volume of data. Auto-encoder model and other deep learning models are compared with the proposed model to evaluate the performance by using a real-time credit card fraud dataset. The accuracy reached is 99%, obtained by the proposed model.

In this Chapter I presented the literature review I conducted to support this thesis, divided into Background and Related Work. In the next Chapter, the Methodology Chapter (3), I show how I designed, and conducted the experiment on this research, and the research design I selected.

Chapter 3

Methodology

3.1 Introduction

This thesis is focused on the practical aspects of machine learning algorithms used in financial fraud detection in real-world networks, and its main feature is the implementation of an approach that aims to mitigate bias on mechanisms, in such a way that keeps the dataset representative of the real-world scenario, despite the preprocessing applied on the dataset.

Mubalaike [18] and Johari [17], applied Restricted Boltzmann Machine (RBM) approach and Convolutional Neural Network (CNN) model, respectively, on the PaySim dataset. They used techniques on the data preparation and modelling. Although their models demonstrated good performances, there were some important considerations missing, and they are:

1. Absence of a validation set, distinct from the test set;

2. The Restricted Boltzmann Machine and Convolutional Neural Network methods used in their studies did not guarantee that all the data (100% of the dataset) was used in both the training and the test phases, meaning that the training data is not used for testing and the test data is not used for training;

3. Limited number of measurement scores used to measure performance.

The relevance of this thesis opens up a discourse on the importance of rigorous validation methods to ensure that synthetic datasets accurately mirror real-world conditions. In essence, while encouraging further exploration in synthetic data generation, it also advocates for a more vigilant and critical perspective on their creation

and validation, fostering a balanced and thoughtful approach to future research in this domain.

This thesis proposes a modified form of the k-fold cross validation method applied to PaySim, which is a synthetic dataset that simulates financial transactions from an African mobile bank. The goals of the application of such approach is to address the challenges of overfitting and inadequate representation of subsets in financial fraud detection datasets. The unbalanced class distribution of fraudulent transactions in these kind of datasets can lead to biased models that fail to detect fraudulent activities. Overfitting can also occur when a model is too complex and fits the training data too closely, resulting in poor performance on unseen data. To mitigate these issues, the proposed technique ensures that each subset maintains the same fraud-to-nonfraud ratio as the entire dataset, confirming representativeness of the subsets in relation to the whole dataset, and also guarantees that all data is used in the training, validation, and test phases, through the use of this modified form of cross validation approach over the PaySim preprocessed dataset.

This thesis applies a range of traditional machine learning techniques, including Random Forest (RF) and Logistic Regression (LR), against a simple neural network with three layers, and compares the results to a prior study that applied a convolutional neural network on the same dataset. The results are promising, indicating that the modified k-fold cross validation method effectively addresses the challenges of overfitting and inadequate representation of subsets in financial fraud detection datasets. The results indicated that my proposed method, Full Sets method, had better scores than the Iterations approach and the machine learning models, and additionally Full Sets method was better than the prior research that it was compared with. This thesis suggests that the techniques I applied can be useful for developing more accurate and reliable fraud detection models in the financial industry.

This Chapter presents the research design used in this thesis and the justification of such choices. It starts with the preprocessing of the dataset, going through the preparation of the preprocessed dataset using a splitting technique that is a modified form of k-fold cross validation and named here as. This approach is compared to the Iterations method, which is a different kind of dataset splitting. Full Sets and Iterations approaches are submitted to a neural network model. Then, the results of the performances of both approaches are compared with each other. The best approach is then compared to the machine learning classifier that best performs among Random Forest, Logistic Regression and AdaBoost, and finally the Full Sets approach

is compared to a prior research that used convolutional neural network over the same PaySim dataset.

3.1.1 Research Design: Justification of Choice

The purpose of this section is to provide a comprehensive justification for the research design choices made in this thesis.

PaySim synthetic dataset

I chose to work with the PaySim synthetic dataset instead of a real-world one for several reasons. Firstly, privacy concerns may arise when working with real-world datasets, as they often contain sensitive or personal information that the dataset owner, financial institutions, may need to protect. This can make it difficult for these companies to release the data to the public. Synthetic datasets, on the other hand, can be designed to mimic realistic data patterns without revealing any private information. Secondly, I believe that synthetic datasets offer researchers a suitable alternative, as they provide a sufficient amount of diverse data and the opportunity to create new patterns that have not yet been studied. Additionally, this does not rely on companies to release their data to the public.

However, This thesis wants to emphasize the importance of dataset selection and critical examination, reminding that model accuracy and effectiveness are not solely the products of the algorithms employed but are also deeply intertwined with the quality and relevance of the dataset used. Therefore, researchers should approach the use of synthetic datasets with caution, thoroughly analyzing their characteristics, generation methods, and relevance to the real-world phenomena they aim to represent.

The Neural Network Model

The model used to evaluate the Iterations and Full Sets approaches is a Keras Sequential model with two layers, dense_148 and dense_149. The dense_148 is a dense layer with 16 units, meaning that it has 16 neurons. The output shape of this layer is (None, 16), where None indicates that the size of the input batch can be of any size, and 16 is the number of output units (neurons).

After dense_148, there is a dropout layer called dropout_74. Dropout is a regularization [1] technique that randomly sets input units to 0 during training, which helps prevent overfitting. This layer has the same output shape as dense_148, which is (None, 16).

The final layer, dense_149, is a dense layer with a single output unit, which means it has one neuron. The output shape of this layer is (None, 1), where None indicates that the size of the input batch can be of any size, and 1 is the number of output units (neurons).

The total number of trainable parameters in the model is 193, which includes 176 parameters in dense_148 and 17 parameters in dense_149. These parameters are the weights and biases of the neural network that are learned during the training process to make predictions on new data.

In Keras, the naming convention for layers is $< layer\_type >\_< number >$. The number after the underscore represents the index of the layer in the model. In this case, dense_148 indicates that this is a Dense layer and it is the 148th layer in the Sequential model. The number 148 is just an arbitrary index assigned by Keras when the layer is added to the model. The index number is used to refer to the layer later in the code if needed, for example, to access the layer's weights or to add a new layer before or after this layer.

Similarly, the naming convention for Dropout layers is $< layer\_type >\_< number >$. The number after the underscore represents the index of the layer in the model. In this case, dropout_74 indicates that this is a Dropout layer and it is the 74th layer in the Sequential model. The number 74 is just an arbitrary index assigned by Keras when the layer is added to the model. The index number is used to refer to the layer later in the code if needed, for example, to access the layer's properties or to add a new layer before or after this layer.

This model architecture is relatively simple and the reason of choosing this algorithm is due to its simplicity, I did not want to have overfitting due to a highly complex model and also I wanted to focus on the splitting technique used in the dataset, called here as modified form of k-fold cross validation approach, which was the core and main subject of study in this thesis.

```
Model: "sequential_74"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_148 (Dense)            (None, 16)                176

 dropout_74 (Dropout)         (None, 16)                0

 dense_149 (Dense)            (None, 1)                 17

=================================================================
Total params: 193
Trainable params: 193
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.1: Layers of the neural network applied.

Dataset preprocessing method

A prior study successfully used the PaySim synthetic dataset, and utilized specific preprocessing techniques for this dataset. I wanted to keep the same pattern of data cleaning, to not bias my comparison results since my focus on the thesis is the using of a modified form of k-fold cross validation method evaluation, through comparison of performances between my proposed approach applied to a three-layer neural network algorithm, and other k-fold splitting approach on the dataset applied to the same algorithm and in sequence a comparison to three machine learning methods.

Iterations approach

This approach centers around the preparation of a dataset post-preprocessing, in which the dataset is bifurcated into two segments: transactions labeled as fraudulent (frauds) comprising 8,231 samples, and those labeled as legitimate (non-frauds) comprising 2,762,200 samples. Subsequently, each of these sets is divided into five subsets, resulting in 1,642 samples for each fraudulent subset and 552,440 samples for each non-fraudulent subset.

Next, five iterations are created by combining one fraudulent subset with one non-fraudulent subset. The result is five distinct subsets, referred to as five iterations, with each representing a 20% fraction of the full dataset. These iterations are designed to be representative of the main dataset due to their preservation of the same class distribution proportionality.

This approach was adopted to explore the possibility of a model performing efficiently across multiple folds, in comparison to the Full Sets approach. A key advantage of this method is that it reduces time, computational power, and cost by a factor of five compared to the variation of the k-fold cross-validation method used in the Full Sets approach. This, in turn, allows for more cost-effective and efficient processing.

The Iterations approach is mapped during the application of the model, as follows:

- Total number of transactions: 554082 (20% of the full dataset).
- Frauds (class 1): 1642 (20% of the total number of fraudulent transactions).
- Nonfrauds (class 0): 552440 (20% of the total number of legitimate transactions).
- Each Iteration is split in Training Set, Validation Set and Test Set.
  - The test set has 30% out of the total data from the full iteration, with 492 fraudulent transactions and 165732 legitimate transactions.
  - The validation set has 20% out of the remaining data from the iteration after subtracting the test set, with 230 fraudulent transactions and 77341 legitimate transactions.
  - The training set has the remaining of the iteration data, with 920 fraudulent transactions and 309367 legitimate transactions.

Diagram: Iterations approach

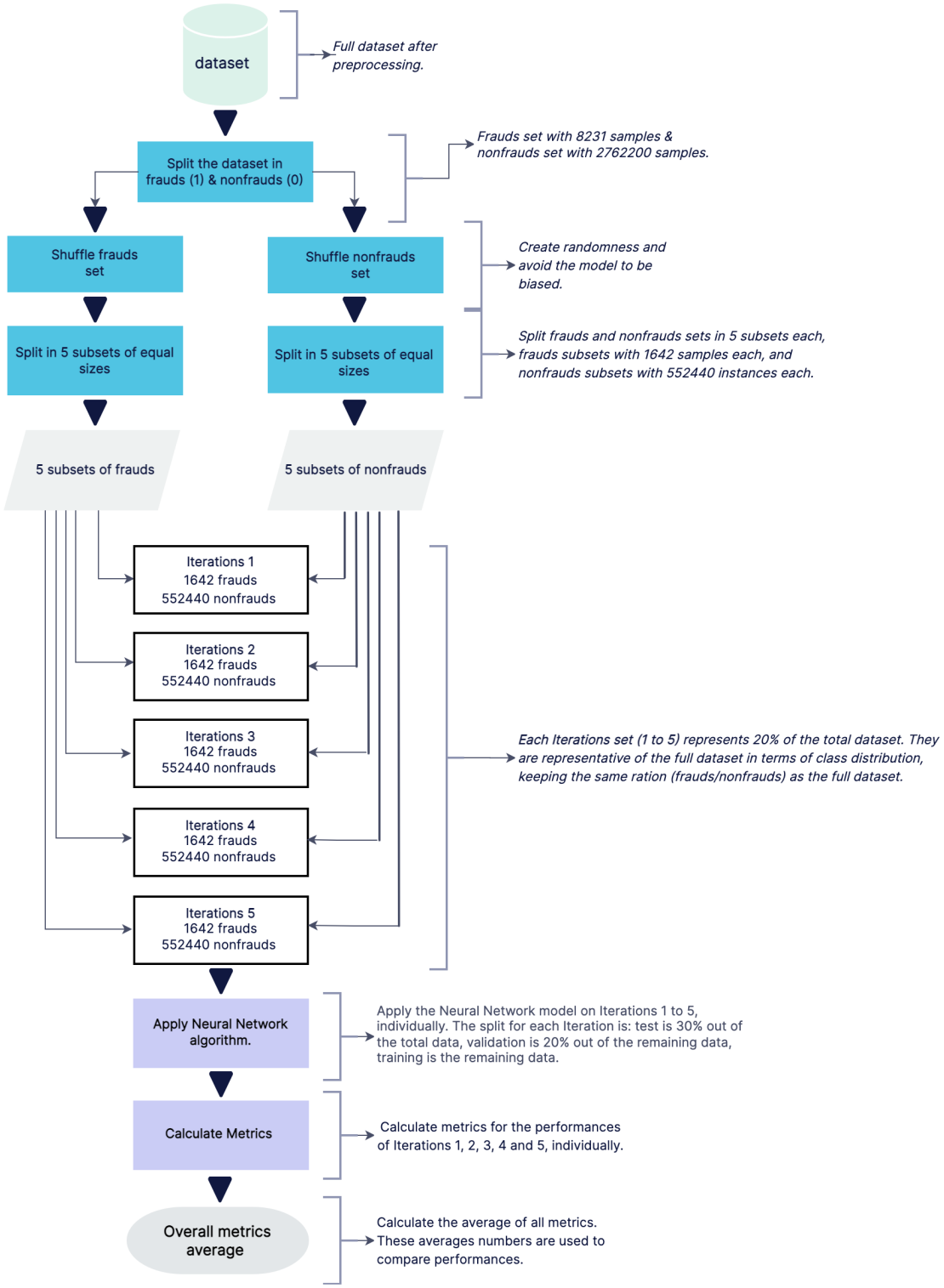

Figure 3.2: Diagram of the Iterations approach process.

Full Sets approach

This approach also starts after the dataset has been preprocessed. The dataset is divided into two groups. One group is for fraudulent transactions, with 8,231 examples, and the other group is for legitimate transactions, with 2,762,200 examples. To keep things random, each subset is shuffled before the splitting in 10 parts. The `.sample(frac=1)` method performs a genuine shuffling of the dataframe's rows. It works by treating the fraction given as the total number of rows to return (1 being all rows), then randomly sampling from the dataframe without replacement, resulting in a shuffled dataframe. The same thing applies for the class 0 (nonfrauds). Listing 3.1 shows the code:

```
df_shuffled = fraud.sample(frac = 1)
df_splits_f = np.array_split(df_shuffled, 10)
df_shuffled_2 = non_fraud.sample(frac = 1)
df_splits_nf = np.array_split(df_shuffled_2, 10)
```

Listing 3.1: Splitting the dataset in frauds and nonfrauds while shuffling

Next, I divide both groups into ten equal parts, or folds. This gives me ten smaller groups of frauds, each with 821 examples, and ten smaller groups of non-frauds, each with 276,220 examples.

To make things clear to understand, I name the fraud groups 'fraud df (x)', where 'x' varies from 'a' to 'j', so I have ten groups in total. Similarly, I name the non-fraud groups 'non-fraud df (y)', where 'y' is a number from 1 to 10, also giving me ten groups in total.

Finally, using these 20 smaller groups - ten frauds and ten non-frauds - I create five datasets that are similar to the full dataset, the Full Sets. Each one of these Full Sets represents a complete dataset, keeping the same ratio frauds/nonfrauds of data as the original dataset.

Each full set is distributed like below.

- Total number of transactions: 2770409 (100% of the dataset).

- Frauds (class 1): 8213 (100% of the total number of fraudulent transactions).
- Nonfrauds (class 0): 2762200 (100% of the total number of legitimate transactions).
- Each full set is split in *Training Set*, *Validation Set* and *Test Set*.
  - The training set has 60% of the full dataset, with 4926 fraudulent transactions and 1657320 legitimate transactions.
  - The validation set has 20% of the full dataset, with 1642 fraudulent transactions and 552440 legitimate transactions.
  - The test set has 20% of the full dataset, with 1642 fraudulent transactions and 552440 legitimate transactions.

The code Listing 3.2 is splitting the dataset into 2 sets, one with only frauds (class 1), with 8231 samples, and another one with only nonfrauds (class 0) with 2762200 instances. Then, these sets are shuffled to create randomness on it, and then generating 10 equal sized dataframes for frauds and 10 for nonfrauds.

```
# split the df in fraud and non-fraud
non_fraud = df[df['isFraud']==0]
fraud = df[df['isFraud']==1]

# split the fraud df into 10 randomly split equal size dataframes
df_shuffled = fraud.sample(frac = 1)
df_splits_f = np.array_split(df_shuffled, 10)

# split the nonfraud df into 10 randomly split equal size dataframes
df_shuffled_2 = nonfraud.sample(frac = 1)
df_splits_nf = np.array_split(df_shuffled_2, 10)
```

Listing 3.2: Creating ten subsets of frauds and ten subsets of nonfrauds

The code in Listing 3.3 is randomly creating the 2 sets of 10 unique integer numbers from 0 to 9 to be used as indices.

```
# Creating 10 random unique integers from the range 0 to 9 (
    inclusive) for two separate lists named selection_1 and
    selection_2.
```

```
2
3 import random
4 selection_1 = random.sample(range(0, 10), 10)
5 selection_2 = random.sample(range(0, 10), 10)
6 selection_1, selection_2
7
8 # Random integers generated.
9 ([4, 8, 5, 0, 6, 1, 2, 9, 3, 7], [4, 0, 9, 1, 6, 3, 7, 8, 2, 5])
```

Listing 3.3: two sets of ten unique integers from 0 to 9

The code in Listing 3.4 is randomly creating the ten subsets for frauds and ten for nonfrauds, using the two sets of ten integers generated before, to apply in the indices of selection_1 (frauds) and selection_2 (nonfrauds).

```
1
2 # After selecting these random indices, the code creates 10 new
    dataframes (fraud_df_a, fraud_df_b, fraud_df_c, ..., fraud_df_j)
    by selecting the split dataframes from df_splits_f dataset based
    on their indices provided in the selection_1 list, and similarly,
     creates 10 new dataframes (nonfraud_df_1, nonfraud_df_2,
    nonfraud_df_3, ..., nonfraud_df_10) by selecting the split
    dataframes from the df_splits_nf dataset based on their indices
    provided in the selection_2 list.
3
4 fraud_df_a = df_splits_f[selection_1[0]]
5 fraud_df_b = df_splits_f[selection_1[1]]
6 fraud_df_c = df_splits_f[selection_1[2]]
7 fraud_df_d = df_splits_f[selection_1[3]]
8 fraud_df_e = df_splits_f[selection_1[4]]
9 fraud_df_f = df_splits_f[selection_1[5]]
10 fraud_df_g = df_splits_f[selection_1[6]]
11 fraud_df_h = df_splits_f[selection_1[7]]
12 fraud_df_i = df_splits_f[selection_1[8]]
13 fraud_df_j = df_splits_f[selection_1[9]]
14
15 nonfraud_df_1 = df_splits_nf[selection_2[0]]
16 nonfraud_df_2 = df_splits_nf[selection_2[1]]
17 nonfraud_df_3 = df_splits_nf[selection_2[2]]
18 nonfraud_df_4 = df_splits_nf[selection_2[3]]
19 nonfraud_df_5 = df_splits_nf[selection_2[4]]
20 nonfraud_df_6 = df_splits_nf[selection_2[5]]
```

```
nonfraud_df_7 = df_splits_nf[selection_2[6]]
nonfraud_df_8 = df_splits_nf[selection_2[7]]
nonfraud_df_9 = df_splits_nf[selection_2[8]]
nonfraud_df_10 = df_splits_nf[selection_2[9]]
```

Listing 3.4: Using the two sets of ten random integers to split the samples into the ten dataframes for frauds and ten for nonfrauds

The code in Listing 3.5 is creating the Training, Validation and Test Sets, split as indicated in the code, concatenating: two fraud subsets and two nonfraud subsets for Test Set, two fraud subsets and two nonfraud subsets for Validation Set, and six fraud subsets and six nonfrauds subsets for Training Set. This process is executed five times, alternating the subsets among the sets and ensuring that the test and validation sets use all fraud subsets from a to j and all nonfraud subsets from 1 to 10, and the training set uses all the fraud and nonfraud subsets as well, during all five rounds.

```

import pandas as pd
fs_test = pd.concat([fraud_df_a, fraud_df_b, nonfraud_df_1,
    nonfraud_df_2], axis = 'rows')
fs_val = pd.concat([fraud_df_c, fraud_df_d, nonfraud_df_3,
    nonfraud_df_4], axis = 'rows')
fs_train = pd.concat([fraud_df_e, fraud_df_f, fraud_df_g, fraud_df_h
    , fraud_df_i, fraud_df_j, nonfraud_df_5, nonfraud_df_6,
    nonfraud_df_7, nonfraud_df_8, nonfraud_df_9, nonfraud_df_10],
    axis = 'rows')
```

Listing 3.5: Creating the Sets (Training/Validation/Test)

The code in Listing 3.6 assigns the Training, Validation and Test Sets to be used by the model during the classifying process. The sample distribution into the *train_df*, *val_df*, and *test_df* are different in all five Full Sets.

```

train_df = fs_train
val_df = fs_val
test_df = fs_test
```

Listing 3.6: Assigning the variables with the sets to be used in the model

Diagram: Full Sets approach

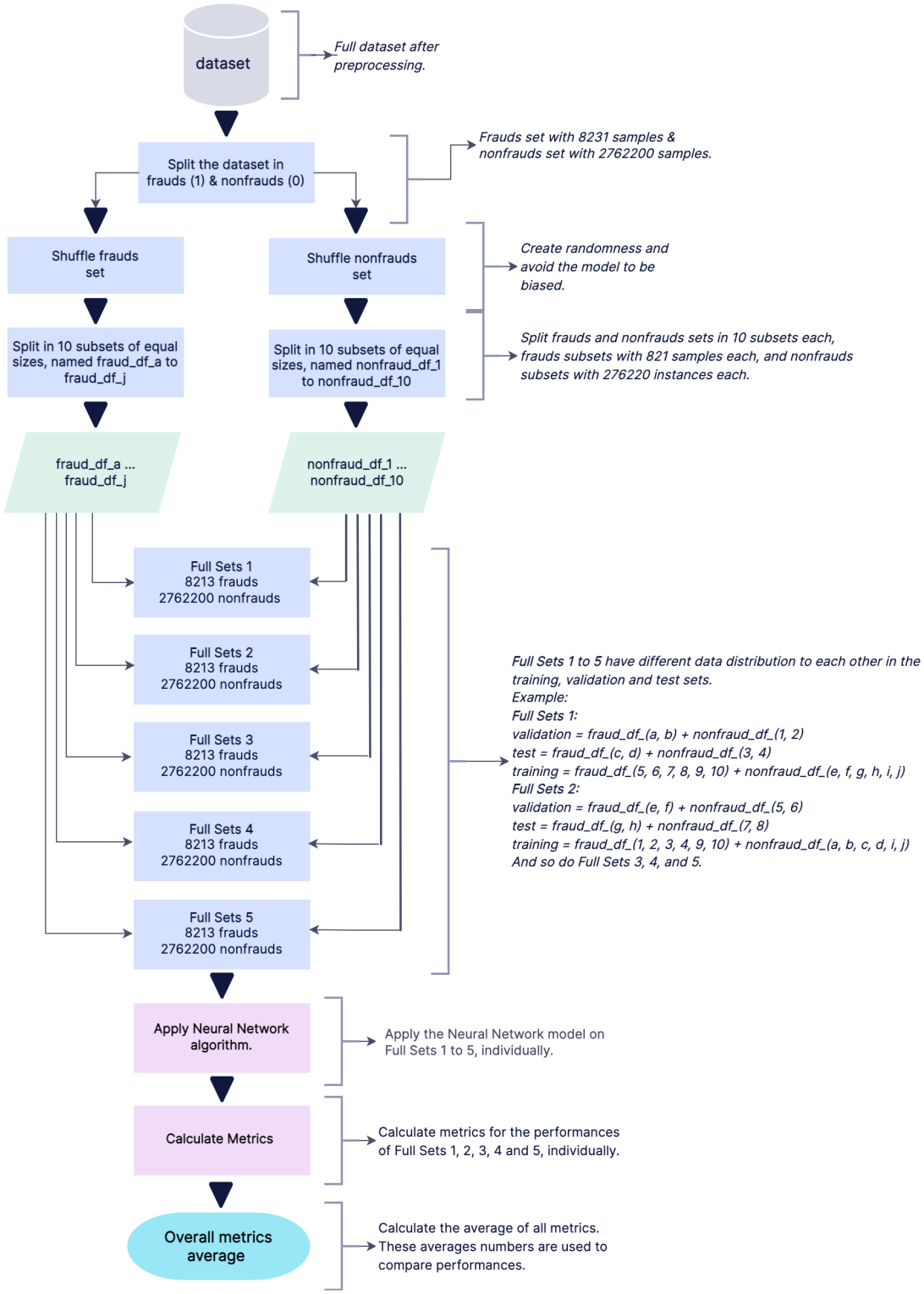

Figure 3.3: Diagram of the Full Sets approach process.

3.1.2 The Graphs

The graphs plotted represent the results of the models performance. There are five graphical curves to support the analysis: Epoch vs Loss, Epoch vs PRC, Epoch vs Precision, Epoch vs Recall, and the ROC curve. The first four graphs have the curves *Train* and *Val* (training and validation), and the fifth graph has the curves *Train Baseline* and *Test Baseline.*

Epoch vs Loss

In the graph of Epoch vs Loss, the x-axis represents the epoch or iteration number, which is a measure of how many times the model has processed the entire training dataset. The y-axis represents the loss, which is a measure of how well the model is able to make predictions on the training data.

During training, the model updates its internal parameters to minimize the loss, which is the difference between the predicted output and the true output. The goal of training is to find the values of the model parameters that result in the lowest loss on the training data.

This graph shows how the loss changes during training, as the model's parameters are updated. The loss normally decreases as the model learns to better fit the training data, but it may eventually plateau or even increase if the model starts to overfit. If the loss decreases rapidly during the initial epochs and then starts to plateau, it may be a sign that the model reached a local minima. On the other hand, if the loss is not decreasing sufficiently, it may be a sign that the model is underfitting [1] the training data and needs to be adjusted.

Epoch vs PRC

The graph Epoch vs PRC (Precision-Recall Curve) with train and validation curves is a plot that shows how the precision and recall change during different epochs in the training of a machine learning model, with separate curves for the training set and validation set.

The Precision-Recall Curve is a commonly used evaluation metric in binary classification tasks. It is a plot of precision versus recall, where precision is the ratio of true positive predictions to the total number of positive predictions, and recall is the ratio of true positive predictions to the total number of actual positive cases in the dataset. The PRC curve shows the trade-off between precision and recall for different thresholds used to classify positive predictions.

In this graph with train and validation curves, the x-axis represents the epochs of the model training, while the y-axis shows the precision and recall values at each epoch. There are two curves, one for the training set and one for the validation set. The training curve shows how the precision and recall values change during training on the training set, while the validation curve shows how the precision and recall values change on a separate validation set that is not used for training.

If both the training precision and recall values increase while the validation precision and recall values also increase or remain stable, it indicates that the model is learning to make more accurate positive predictions and is not overfitting the training data. However, if the training precision and recall values increase while the validation precision and recall values decrease, it suggests that the model may be overfitting the training data and not generalizing well to new data.

Epoch vs Precision

The graph Epoch vs Precision with train and validation curves is a plot that shows how the precision metric changes during different epochs in the training of a machine learning model.

In this graph with train and validation curves, the x-axis represents the epochs of the model training, while the y-axis shows the precision values at each epoch. There are two curves, one for the training set and one for the validation set. The training curve shows how the precision values change during training on the training set, while the validation curve shows how the precision values change on a separate validation set that is not used for training.

If the training precision values increase while the validation precision values also increase or remain stable, it indicates that the model is learning to make more accurate positive predictions and is not overfitting the training data. However, if the training precision values increase while the validation precision values decrease, it suggests

that the model may be overfitting the training data and not generalizing well to new data.

This graph can also help to determine where you could stop training a model, for example, when the validation precision values reach a plateau or begin to decrease, so you can run again the model with enough number of epochs just to reach that point in the curve.

Epoch vs Recall

The graph Epoch vs Recall with train and validation curves is a plot that shows how the recall metric changes in different epochs during the training of a machine learning model.

In this graph with train and validation curves, the x-axis represents the epochs of the model training, while the y-axis shows the recall values at each epoch. There are two curves, one for the training set and one for the validation set. The training curve shows how the recall values change during training on the training set, while the validation curve shows how the recall values change on a separate validation set that is not used for training.

If the training recall values increase while the validation recall values also increase or remain stable, it indicates that the model is learning to identify more of the actual positive cases in both the training and validation sets. However, if the training recall values increase while the validation recall values decrease, it suggests that the model may be overfitting the training data and not generalizing well to new data.

3.2 Dataset

3.2.1 PaySim Dataset: Features

See how PaySim dataset is structured in terms of features (columns) and their description:

- **step** - maps a unit of time in the real world (e.g., step 1 is 1 hour of time duration and it includes all transactions during this time). The total steps are 744 in 30 days simulation.
- **type** - different kinds of transactions, CASH IN, CASH OUT, DEBIT, PAYMENT, and TRANSFER.
- **amount** - amount of the transaction in local currency.
- **nameOrig** - customer who started the transaction.
- **oldbalanceOrg** - initial balance before the transaction.
- **newbalanceOrig** - new balance after the transaction.
- **nameDest** - customer who is the recipient of the transaction.
- **oldbalanceDest** - new balance recipient after the transaction.
- **isFraud** - This feature represents the class in which indicates whether it is a fraud or legitimate transaction. The agents inside the simulation perform the transactions. In this specific dataset, the fraudulent behaviour of the agents aims to profit by taking control of customers' account and try to empty the funds by transferring to another account and then cashing out of the system.
- **isFlaggedFraud** - the model aims to control high money transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0

Figure 3.4: PaySim raw dataset features (head)

3.2.2 PaySim dataset preprocessing

The techniques applied to the PaySim dataset to treat the raw dataset and transform into a clean one in order to help the model perform efficiently and have optimum results, are: *feature selection (parts A and B), feature extraction, verification of missing values and empty cells, one hot encoding, and feature scaling (standardization).* The description follows.

Figure 3.5 below shows the raw dataset head with all originals features.

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0

Figure 3.5: PaySim dataset in a raw format

Feature selection (part A): removing irrelevant features

Feature selection is the first phase to be applied in the preprocessing of the dataset, used to remove the irrelevant features, i.e., features that will not have any effect during the application of the model and the results. The PaySim dataset in a raw format, has 6362620 rows and 11 columns (features). Figure 3.5 displays the head of the dataset with all the original features included. The features removed are "nameOrig", "nameDest", and "isFlaggedFraud". Due to the nature of this thesis which is focused on the legitimate and fraudulent transactions, "nameOrig" and "nameDest" have no impact on the results, they represent information about the the origin of the transaction and its destination. "IsFlaggedFraud" occurs just 16 times and "isFraud" is set for each occurrence in which "isFlaggedFraud" is flagged so it is redundant to keep the feature, so I removed. Figure 3.6 represents the remaining features in the dataset, after feature selection. Following the removal of irrelevant features, there are 6362620 rows $\times$ 8 columns (features).

step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
1	PAYMENT	9839.64	170136.00	160296.36	0.00	0.00	0
1	PAYMENT	1864.28	21249.00	19384.72	0.00	0.00	0
1	TRANSFER	181.00	181.00	0.00	0.00	0.00	1
1	CASH_OUT	181.00	181.00	0.00	21182.00	0.00	1
1	PAYMENT	11668.14	41554.00	29885.86	0.00	0.00	0

Figure 3.6: Dataset after feature selection (part A)

Feature selection (part B): removing irrelevant categories from specific feature

A dataset is subject to the possibility of overfitting where the model fits the noise in the data rather than the underlying pattern. By selecting only the most relevant features, we can reduce the risk of overfitting. Also by selecting the most relevant features, the performance of the model can be improved by reducing the amount of noise in the data. Another relevant consideration is that having fewer features can also reduce the computational cost of training a model and making predictions. For these reasons, it is desirable to apply the technique of *feature selection*.

The *type* feature represents the kinds of financial transactions, for PaySim there are five kinds: CASH IN, CASH OUT, DEBIT, PAYMENT, and TRANSFER. However, after analysis of the data, I inferred that fraudulent transactions only occur in two types, TRANSFER and CASH OUT. CASH IN, DEBIT, and PAYMENT do not present any fraudulent transactions, therefore they are not considered and are removed. The remaining types of transactions considered in the dataset are CASH OUT and TRANSFER. Figure 3.7 shows the new set. There are 2770409 rows $\times$ 8 columns (features), after feature selection (part B).

step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
1	TRANSFER	181.00	181.00	0.0	0.00	0.00	1
1	CASH_OUT	181.00	181.00	0.0	21182.00	0.00	1
1	CASH_OUT	229133.94	15325.00	0.0	5083.00	51513.44	0
1	TRANSFER	215310.30	705.00	0.0	22425.00	0.00	0
1	TRANSFER	311685.89	10835.00	0.0	6267.00	2719172.89	0

Figure 3.7: Dataset after feature selection (part B)

Performing Missing Data Detection

A dataset originated from a real-world financial network, is constantly subject to inconsistencies such as systems that are electrically unstable generating null data, or missing information. Therefore, a full verification on this dataset is necessary to check whether this dataset has an empty cell or missing value, to prevent inconsistent data from affecting performance of the model. Consequently, verification of all cells is performed with the aim of detection of an empty cell. Once an empty cell has been moved to a specific line, the full transaction is removed. Only transactions presenting all the features with valid information are considered.

The verification for occurrence of any empty cell is done through this line of Python code `iteration_one.isnull().sum()` which calculates the sum of null values for each feature.

This is the result:

```
step                0
amount              0
oldbalanceOrg       0
newbalanceOrig      0
oldbalanceDest      0
newbalanceDest      0
isFraud             0
CASH_OUT            0
TRANSFER            0
errorbalanceOrig    0
errorbalanceDest    0
```

There are no empty cells. There are 2770409 rows $\times$ 8 columns (features), after this step.

One hot encoding

One hot encoding is a technique used in machine learning to represent categorical variables, which are normally alphabetical or alphanumerical, as numerical data. It is useful in classification tasks where the model interprets numbers, classifying the classes into 0 or 1.

For this dataset, the feature called "type" was represented by two values, *cash out* and *transfer*. So, to use one hot encode in this variable, two binary vectors were

created: [1, 0] for *cash out* and [0, 1] for *transfer*. Table 3.1 represents this numerical categorical feature in the dataset:

Table 3.1: One-Hot Encoding of Transaction Types

Types of transaction	**Is_CASH_OUT**	**Is_TRANSFER**
CASH_OUT	1	0
TRANSFER	0	1

Figure 3.8 displays the new format of the dataset with the features after the application of one hot encoding technique. The categorical feature *type* (cash_out and transfer, the binary vector) is merged with the dataset.

step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	CASH_OUT	TRANSFER
1	181.00	181.00	0.0	0.00	0.00	1	0	1
1	181.00	181.00	0.0	21182.00	0.00	1	1	0
1	229133.94	15325.00	0.0	5083.00	51513.44	0	1	0
1	215310.30	705.00	0.0	22425.00	0.00	0	0	1
1	311685.89	10835.00	0.0	6267.00	2719172.89	0	0	1

Figure 3.8: Dataset updated after one hot encoding

Feature extraction

Feature extraction in a dataset is also called feature engineering, and it is the process of transforming or combining existing features to create new ones that may be more informative for a particular task. In the case of this thesis, the technique used is creating two new features through the combination of three other existing ones. These new features are *errorbalanceOrig* and *errorbalanceDest* and their functions are to differentiate between fraudulent and genuine transactions and to record errors in the originating and destination accounts for each transaction. The following mathematical equations combine *newbalanceOrig* and *amount* in the first new feature

(errorbalanceOrig), and *oldbalanceDest*, *amount*, and *newbalanceDest* in the second new feature (errorbalanceDest). The dataset has 2770409 rows × 11 columns (features) after feature extraction.

$$\text{errorbalanceOrig} = \text{oldbalanceOrig} - \text{amount} - \text{newbalanceOrig} \tag{3.1}$$

$$\text{errorbalanceDest} = \text{oldbalanceDest} + \text{amount} - \text{newbalanceDest} \tag{3.2}$$

Figure 3.9 presents the dataset with the new features included:

step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	CASH_OUT	TRANSFER	errorbalanceOrig	errorbalanceDest
1	181.00	181.00	0.00	0.00	0.00	1	0	1	181.00	181.00
1	181.00	181.00	0.00	21182.00	0.00	1	1	0	181.00	21363.00
1	229133.94	15325.00	0.00	5083.00	51513.44	0	1	0	229133.94	182703.50
1	215310.30	705.00	0.00	22425.00	0.00	0	0	1	215310.30	237735.30
1	311685.89	10835.00	0.00	6267.00	2719172.89	0	0	1	311685.89	-2401220.00

Figure 3.9: Dataset including the new feafures

Feature scaling (standardization)

Feature scaling, also known as standardization or normalization, is a technique used in machine learning to re-scale the feature values in a dataset to a common scale. The purpose of normalization is to ensure that all features contribute equally to the analysis and prevent features with large values from dominating the model.

Min-Max scaling is the feature scaling technique used in this dataset. In the Min-Max scaling technique, the minimum and maximum values of a feature are used to re-scale the values to a range between 0 and 1. The following mathematical formula is used to re-scale the values.

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{3.3}$$

Using the formula 3.3, X is the original value of the variable, $Xmin$ is the minimum value of the variable in the dataset, $Xmax$ is the maximum value of the variable in the dataset, and X' is the normalized value of the variable. The resulting X' value will be between 0 and 1, inclusive, representing the relative position of the original value within the range of the dataset.

Diagram of the Dataset Preprocessing

Figure 3.10 displays a diagram which represents the process of preprocessing applied to the PaySim dataset in its raw format.

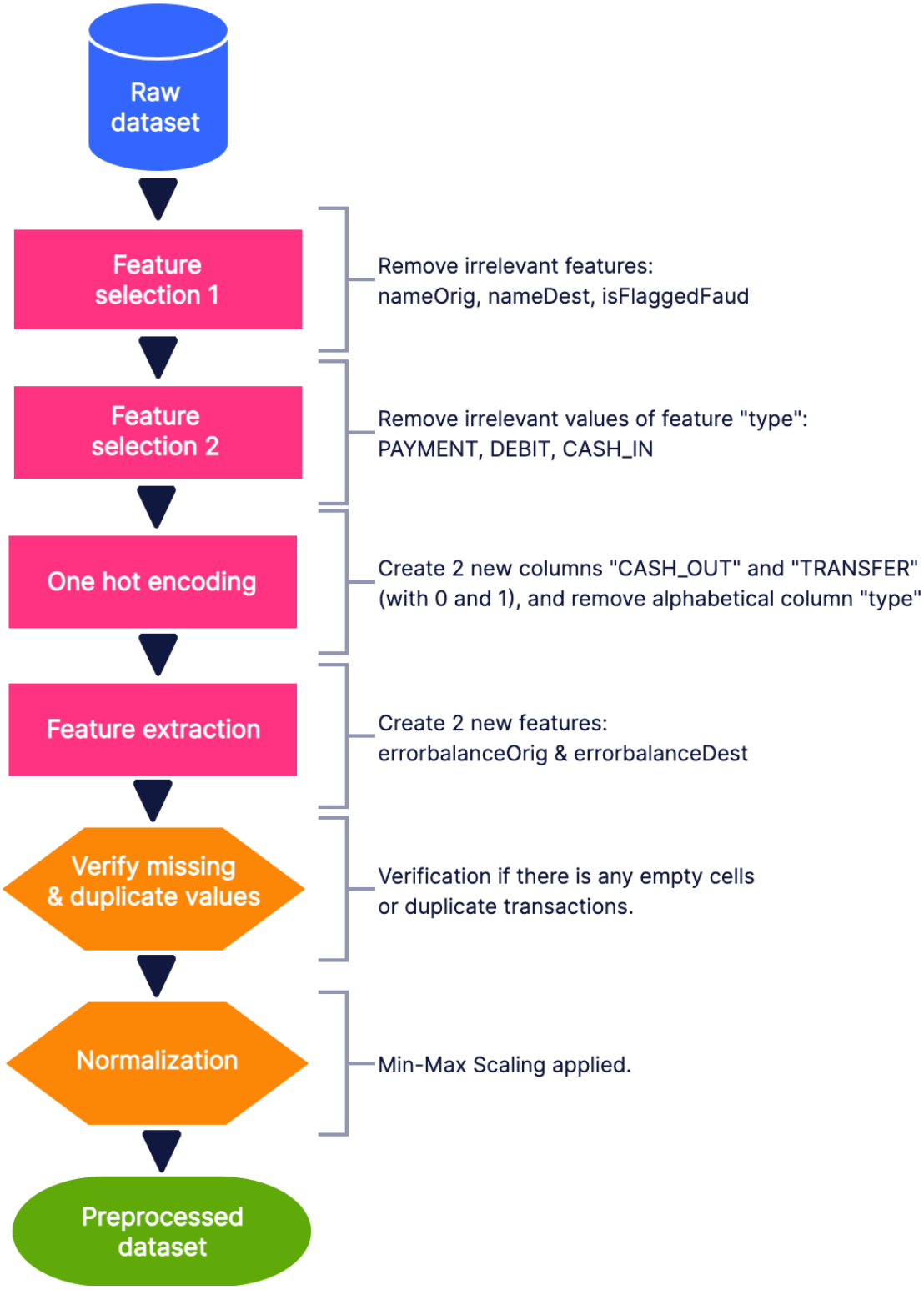

Figure 3.10: Preprocessing process on PaySim dataset

3.3 The Models

With the completion of the preprocessing stage, the dataset is now ready. The subsequent phase involves the application of the classification models, including a neural network for both Iterations and Full Sets approaches, and also Random Forest, Logistic Regression, and AdaBoost. Following the modeling, I will conduct performances analysis and comparisons to evaluate each model's effectiveness.

Figure 3.11 shows how the data frames fraud (class 1) and nonfraud (class 0) are randomly distributed in 10 equal parts, and then a combination of two sets of class 0 and two sets of class 1 form an iteration.

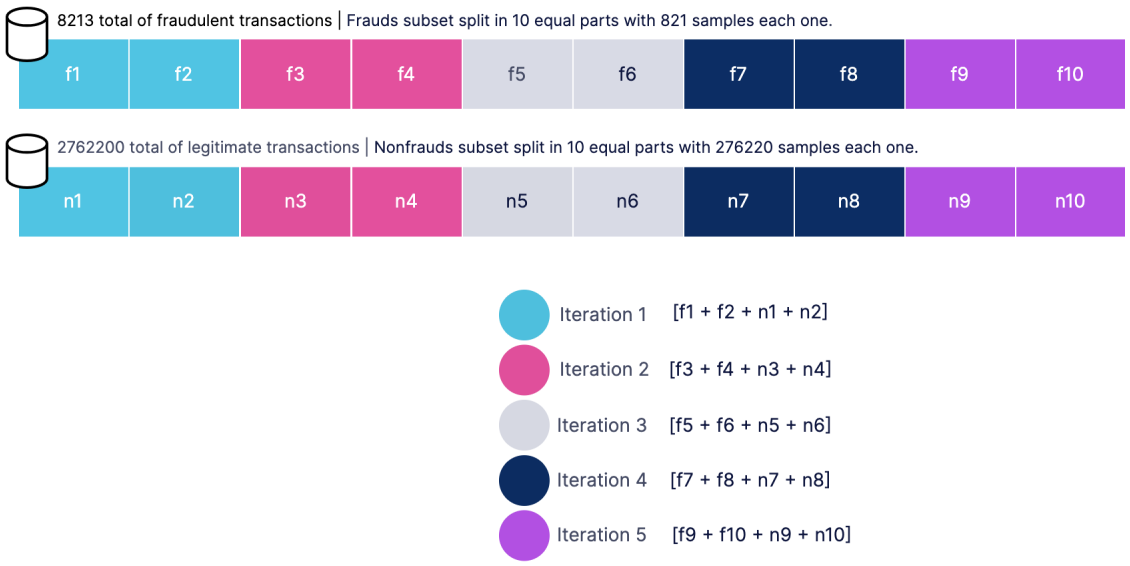

Figure 3.11: Diagram of how the dataset was randomly and equally distributed in 5 representative sets

Detailing each iteration:

- It has 1642 frauds, class 1, and 5522440 nonfrauds, class 0.
- It is representative of the full dataset.
- Classes 0 & 1 are equally distributed, proportionally to the full dataset.

- It is randomly and equally split in each model application.

3.3.1 Running the Models

Now that the dataset is split in five representative sets, it is possible to proceed with the four phases of the proposed approach, specifically:

1. Apply Neural Network Model five times, each one using a separate set (iterations) as the training, validation and test subsets, and extract the average of all the measurement scores collected from the algorithm, for each time the model is run.

2. Apply Neural Network Model five times, each one using all the data (full dataset), splitting six subsets (10% of the dataset each subset, f1 to f10 for frauds, n1 to n10 for nonfrauds, Figure 3.11) for training, two for validation and two for testing, and extract the average of the measurements results.

3. Apply Random Forest classifier, Logistic Regression classifier, and AdaBoost classifier on the full dataset and compare the results with results of the Full Sets scores average, since it involves the full dataset.

4. Compare Full Sets approach with a prior research that used Convolutional Neural Network on the same PaySim synthetic dataset.

The scores used in this thesis where the evaluation are based on, are true positives (TP), false positives (FP), true negatives (TN), false negatives (FN), accuracy (ACC), precision, recall (or sensitivity), area under the curve (AUC), precision-recall curve (PRC), specificity, and f1-score.

3.4 Ethics

In this section I present the ethical considerations of my thesis. As my thesis deals with a binary classification problem applied to a financial synthetic dataset, there are

several ethical aspects I should consider while conducting my research and writing this thesis.

Firstly, my research is based on synthetic financial datasets. I should acknowledge that while synthetic data is beneficial in safeguarding privacy, the process of generating such data needs to be totally transparent and accountable. It is also important to consider the potential limitations and biases that could exist in synthetic data, particularly concerning the ability of this dataset to truly represent real-world scenario and whether it can potentially introduce bias into the machine learning and deep learning algorithms applied.

Secondly, the use of machine learning and deep learning algorithms demands a discussion on fairness, accountability, transparency, and explainability. I must ensure that the algorithms do not perpetuate existing biases, whether from the data or the ways in which the algorithms themselves are constructed and optimized. I must also address the issue of the *black box problem*[1] often associated with these algorithms, as it relates to the ethical responsibility of providing clear, understandable explanations of how our models work and make decisions.

Thirdly, in comparing the performance of different methodologies, including my proposed method with prior research, I should ensure that these comparisons are fair, objective, and transparently reported. Any potential conflicts of interest, funding sources, or affiliations that could influence the interpretation of these comparisons must be openly declared.

Lastly, while my research may not directly involve human subjects, the outcomes could potentially influence financial decisions and policies. Therefore, I must consider the potential impacts in the society and consequences of my research, such as potential misuse of my findings or effects on economic disparities. My commitment to responsible AI includes the ongoing evaluation of these broader impacts and my readiness to engage in dialogue and action to mitigate potential harm.

These ethical considerations serve as a guide for maintaining integrity and responsibility throughout my research process.

This section closes the Methodology Chapter where I presented the methodology

[1]Black Box problem refers to a type of system where the internal mechanisms are not understood or interpretable by humans, even though the input and output are observable.

employed and the research techniques of my thesis. The next Chapter is the Results Analysis, where I highlight my findings and present the analysis to address the research questions and test the hypotheses as the direct outcome of my methodology design.

Chapter 4

Results Analysis

The Results Analysis Chapter of this thesis provides a comprehensive exploration of the performance of the neural network algorithm used with a binary classification problem within a specific financial dataset, the PaySim dataset.

The Results Analysis chapter of this thesis presents an examination of the performance of a neural network algorithm applied to a binary classification problem a financial institution datasets, the PaySim dataset. The chapter goes through the analysis of the results obtained from the experiment, which involved the application of the neural network algorithm to the dataset, in two types of data preprocessing formats for the PaySim dataset, as explained in Chapter 3.

The purpose of this chapter is to provide an understanding of the effectiveness of data preparation in terms of splitting the data, representativeness, proportionality of the classes and the distribution in the subsets, when the neural network algorithm is applied, and the results are compared with to the machine learning approaches Random Forest, Logistic Regression, and AdaBoost. By solving the binary classification problem in the financial bank dataset, we compare the measurement scores, giving more focus to the confusion matrix [31]: true positives, true negatives, false positives, and false negatives.

The chapter also presents graphical representations of the results. These analyses aim to provide insight into the strengths and weaknesses of the approaches used in this thesis.

The insights and findings can be used to guide future research in the area of machine learning applied to financial data analysis.

4.1 Base Rate Fallacy

In the context of a financial dataset where fraudulent transactions constitute a small minority, the base rate fallacy [44] can be particularly relevant. The base rate fallacy, or base rate bias, refers to the cognitive bias where individuals ignore or underestimate the base rate information (prior probabilities) when evaluating the probabilities of outcomes.

In the Full Sets approach, the test set has 552440 non-fraudulent transactions and 1642 fraudulent ones. Hence, the base rate or prior probability of a transaction being fraudulent is 1642 / (552440 + 1642) = 0.003 or approximately 0.3%. This indicates that, in the absence of any other information, any randomly picked transaction has a 0.3% chance of being fraudulent.

Considering a hypothetical scenario where a classification model has been trained on this dataset and it has a high accuracy of 99%, while this might sound impressive initially, considering the base rate, this might not actually be as good as it sounds due to the unbalance in the dataset.

If this model flags a transaction as fraudulent, based on its high accuracy one might think that there's a high probability of this transaction actually being fraudulent. However, this can be misleading due to the base rate fallacy. Given the low base rate of fraud (0.3%), a large number of non-fraudulent transactions might still be incorrectly classified as fraudulent (false positives), especially when the total number of non-fraudulent transactions is very large.

4.2 Results: Iterations approach

In this section, the results of the dataset split into five sets equally distributed with each submitted to the neural network algorithm. Table 4.1 presents the results with the average also calculated and displayed. An average number of the results are also calculated and presented.

Table 4.1: Results of the five iterations (each with 20% of dataset) scores and the average

Scores	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Average
loss	0.00334	0.00300	0.00405	0.00341	0.00478	0.00371
true positives	**349**	**350**	**354**	**349**	**308**	**342**
false positives	**7**	**7**	**4**	**8**	**4**	**6**
true negatives	165746	165729	165725	165713	165747	165732
false negatives	**123**	**139**	**142**	**155**	**166**	**145**
accuracy	0.99922	0.99912	0.99912	0.99902	0.99898	0.99909
precision	0.98034	0.98039	0.98883	0.97759	0.98718	0.98287
recall or sensitivity	0.73941	0.71575	0.71371	0.69246	0.64979	0.70222
AUC	0.98099	0.99624	0.98498	0.99120	0.96737	0.98416
PRC	0.87847	0.90886	0.90128	0.88562	0.86381	0.88761
specificity	0.99996	0.99996	0.99998	0.99995	0.99998	0.99996
f1 score	0.84299	0.82742	0.82903	0.81068	0.78370	0.81876
Total transactions	166225.0	166225.0	166225.0	166225.0	166225.0	

Calculation of the *Average*:

$$\text{Average} = \frac{\text{Iteration1} + \text{Iteration2} + \text{Iteration3} + \text{Iteration4} + \text{Iteration5}}{5} \tag{4.1}$$

Each iteration is distributed as follows.

- Total number of transactions: 554082 (20% of the full dataset).
- Frauds (class 1): 1642 (20% of the total number of fraudulent transactions).
- Nonfrauds (class 0): 552440 (20% of the total number of legitimate transactions).
- Each *Iteration* is split in *Training Set*, *Validation Set* and *Test Set*.
  - The test set has 30% of the full iteration, with 492 fraudulent transactions and 165732 legitimate transactions.
  - The validation set has 20% out of the remaining iteration after deducted the test set, with 230 fraudulent transactions and 77341 legitimate transactions.
  - The training set has the remaining of the iteration data, with 920 fraudulent transactions and 309367 legitimate transactions.

4.2.1 Iteration Analysis (Average)

- True Positives (TP): This refers to fraudulent transactions that were correctly identified by the model as fraud. There are 342 true positives, meaning the model successfully identified 342 fraudulent transactions.

- False Positives (FP): These are legitimate transactions that the model incorrectly flagged as fraudulent. In this case, there are 6 false positives, which is quite low, indicating the model is precise in its positive (fraud) classifications. However, this could cause inconvenience for customers whose transactions were incorrectly flagged.

- False Negatives (FN): These are fraudulent transactions that the model incorrectly identified as legitimate. There are 145 false negatives, meaning 145 fraudulent transactions went undetected by the model. This is more concerning, as it represents missed fraudulent activities and potential security risks.

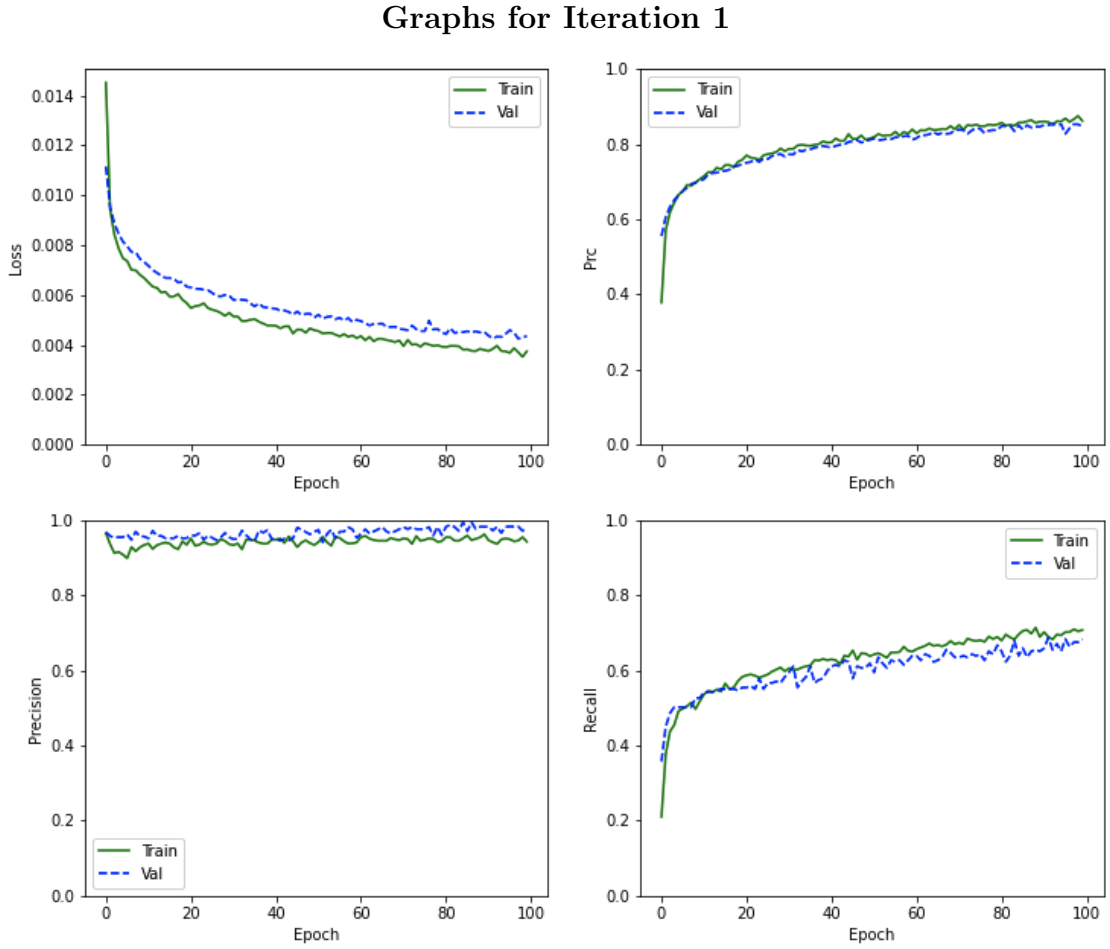

Figure 4.1: Graphs: Loss vs Epoch, PRC (Precision-Recall Curve) vs Epoch, Precision vs Epoch, and Recall vs Epoch for Iteration 1

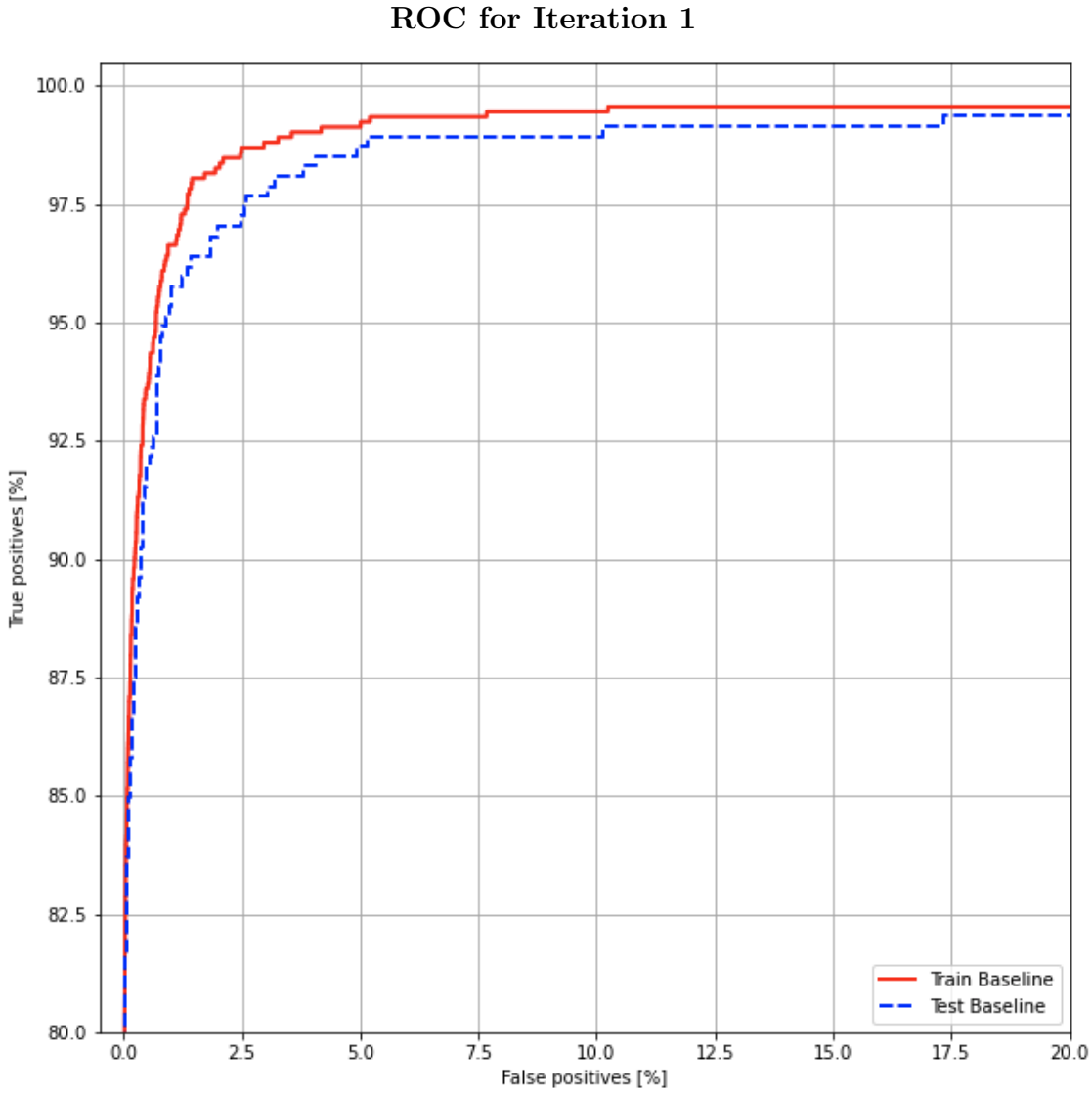

Figure 4.2: ROC for Iteration 1

4.2.2 Why the results from the Iterations differ from each other?

The variation in the results is a common occurrence when splitting data and applying machine learning models, and it can be attributed to some factors. In the case of this thesis, I applied the neural network algorithm. The possible reasons for these variations are:

- *Variability in the data splits:* When I randomly split the PaySim financial dataset into five parts, each split contains a different subset of the data. Since every split is unique, the neural network will be learning from slightly different information each time. Consequently, the model might perform better on some splits than others due to the particular characteristics of the data in that split.
- *Model Initialization:* Neural networks typically initialize their weights and biases randomly. This means that even if you train the exact same model architecture on the exact same data, you might get different results simply due to the randomness in this initialization.

These factors are part of the reason why I use this approach of splitting the dataset and calculate an average of the five splits, which can provide a more robust estimate of the model's performance by averaging the results from multiple splits of the data.

4.2.3 Graph Analysis: Iterations approach

This is the graph section of the 5 iterations results from the Iterations approach. There are 5 graphs for each model application as seen previously in the graph results of each iteration, and they are: Loss vs Epoch, PRC vs Epoch, Precision vs Epoch, Recall vs Epoch, and True positives [%] vs False positives [%]. For the first 4 graphs, the curves are *Train* and *Validation*, and for the last graph, the curves are *Train* and *Test*.

See the analysis for each one.

1. **Graph Loss vs Epoch:**

   This Loss vs Epoch graph shows both curves having the same pattern, it indicates that both the training and validation sets are performing similarly throughout the epochs. This can be a good indication that the neural network model is not overfitting or underfitting the data. Overfitting would typically manifest as the training loss continuing to decrease as the model learns, while the validation loss decreases up to a point, but then starts to increase. This increase in validation loss is due to the model beginning to fit the noise in the training data, which does not generalize well to the validation set, while underfitting would often be represented by both the training and validation loss decreasing very slowly, or even plateauing at a high level. This suggests that the model is not able to learn from the data effectively, and this way cannot reduce its prediction error (loss). If the loss curves for both sets have the same pattern, it indicates that the model is generalizing well to the unseen data in the validation set.

   Figure 4.3 shows this graph for Iteration 1.

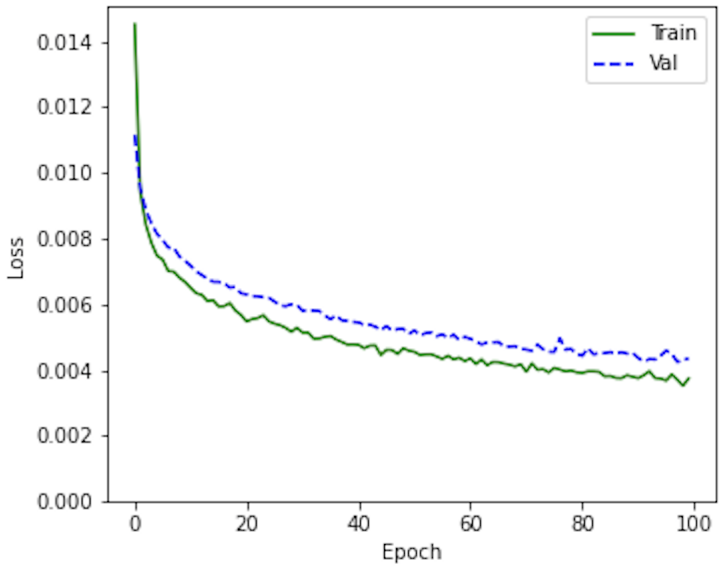

Figure 4.3: Graph Loss vs Epoch for Iteration 1

2. **Graph PRC vs Epoch:**

   From the outset (first epoch), the training and validation curves parallel each other, consistently exhibiting improved performance through decreasing loss (error) with the increment of epochs. This suggests that the model effectively

avoids overfitting to the training data and shows its capacity to generalize to new data. Besides, it indicates the model's ability to successfully capture the underlying patterns of the data.

Figure 4.4 shows this graph for Iteration 1.

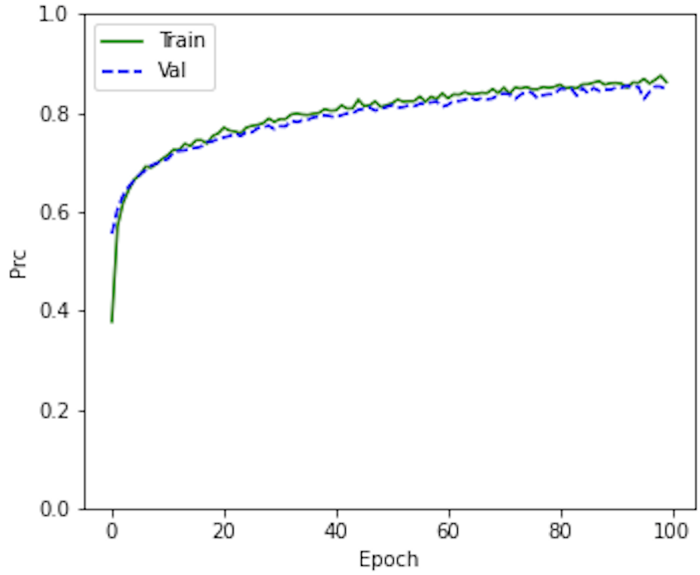

Figure 4.4: Graph PRC vs Epoch for Iteration 1

3. **Graph Precision vs Epoch:**

   Both training and validation curves starts from a high value of 0.9 and maintain a high value up to around 0.99, therefore it suggests that the model is performing extremely well on both sets. Specifically, it means that, from the beginning of the training process (first epoch), the model is able to accurately identify positive instances, with a very small fraction of its positive predictions being incorrect. As the number of epochs increases and the precision stays high or even improves, this suggests that the model continues to maintain its performance in terms of minimizing false positive predictions. This is a very positive sign, as it indicates that the model is not only able to learn the underlying patterns in the data, but is also able to generalize very well to new data, meaning that in this case, the model is running correctly to identify the majority of the positive cases.

   Figure 4.5 shows this graph for Iteration 1.

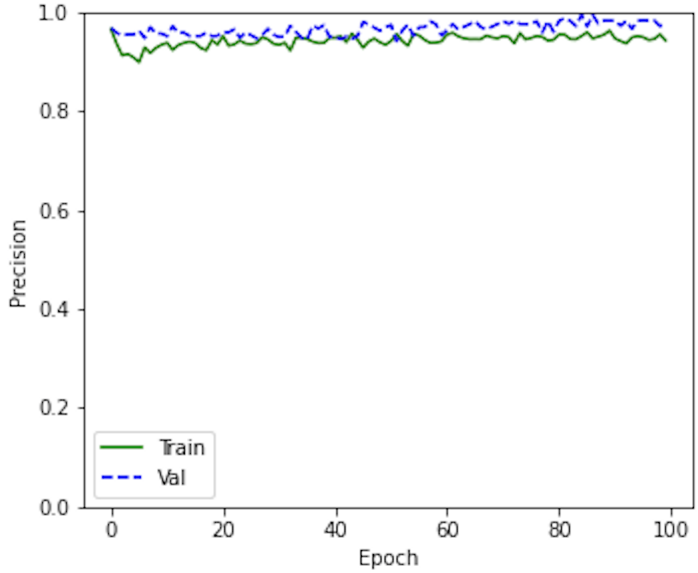

Figure 4.5: Graph Precision vs Epoch for Iteration 1

4. **Graph Recall vs Epoch:**

   This graph shows that both the training and validation curves start from a value of 0.2 and rise together up to a value of around 0.7, while maintaining similar patterns. When both the training and validation curves rise together and maintain similar patterns, it means that the model is performing consistently well on both sets, and in this case, the neural network model is being able to learn well about the positive cases, understanding the underlying patterns in the data and being able to generalize well to unseen data.

   Figure 4.6 shows this graph for Iteration 1.

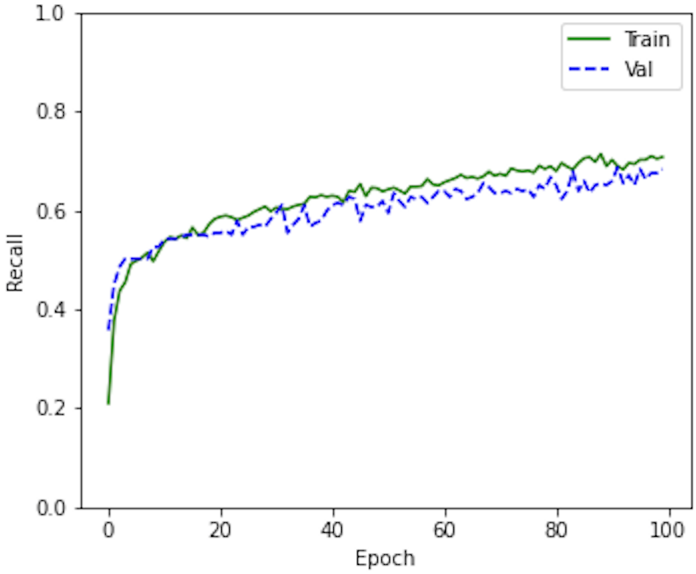

Figure 4.6: Graph Recall vs Epoch for Iteration 1

See analysis for ROC curve graphs (True positives [%] vs False positives [%]):

Figure 4.7 is a graph for the ROC curve commonly used for evaluating the performance of a binary classifier, it plots the true positive rate against the false positive rate as the decision threshold for classification is varied.

In general, a good classifier will have a ROC curve that is as close to the top left corner of the graph (representing high true positive rate and low false positive rate) as possible. Although the curve is near the top left of the graph, meaning that it performs well, there is still room for improvement.

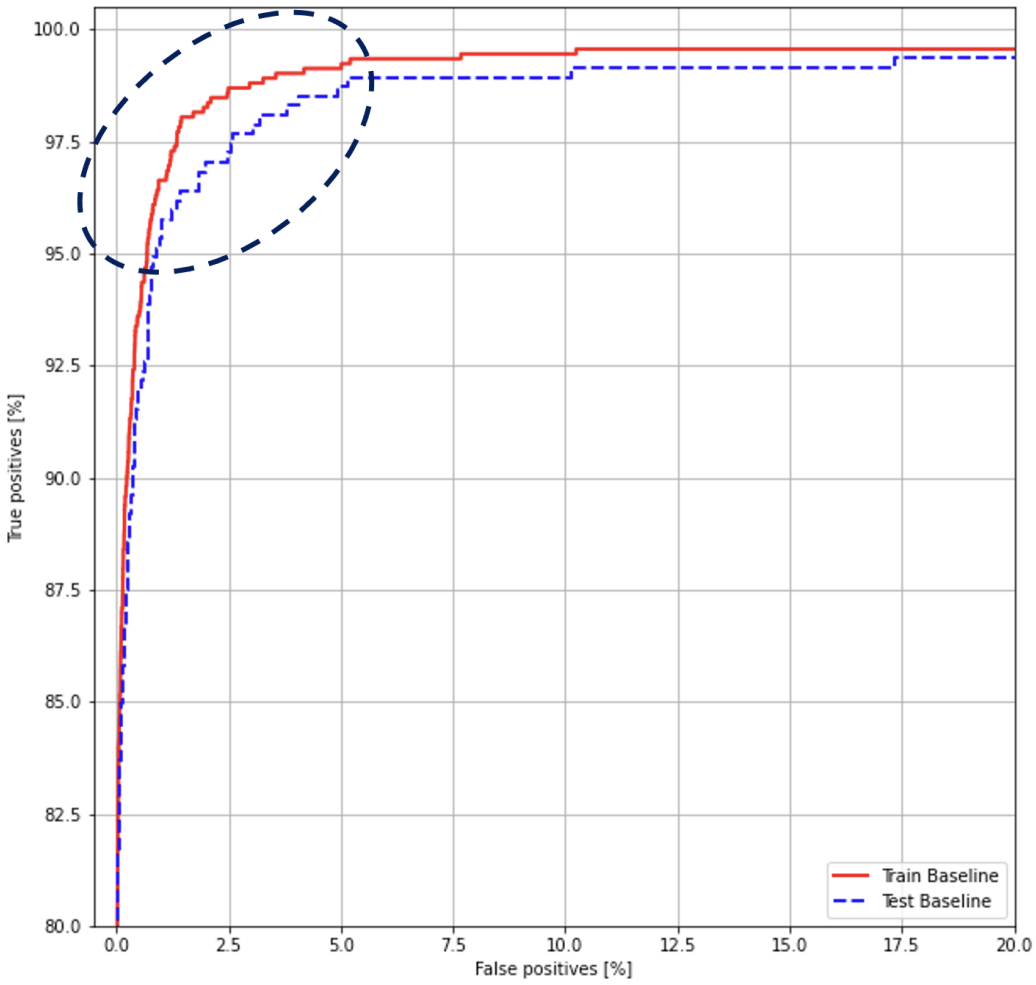

Figure 4.7: Graph True Positives rate vs False Positives rate (ROC curve) for Iteration 1

4.3 Results: Full Sets approach

In this section, the results of the Full Sets approach representing each one the full dataset in the model application are presented. Each set is submitted through the neural network algorithm proposed. Table 4.2 presents the measurements results and the average number for all five sets.

Table 4.2: Measurements for the five Full Sets and average number for each score

Scores	FullSet1	FullSet2	FullSet3	FullSet4	FullSet5	Average
loss	0.00217	0.00279	0.00262	0.00264	0.00269	0.00258
true positives	**1313**	**1126**	**1183**	**1251**	**1195**	**1214**
false positives	**49**	**7**	**9**	**20**	**9**	**19**
true negatives	552391	552431	552431	552418	552431	552420
false negatives	**329**	**517**	**459**	**393**	**447**	**429**
accuracy	0.99932	0.99905	0.99916	0.99925	0.99918	0.99919
precision	0.96402	0.99382	0.99245	0.98426	0.99252	0.98542
recall or sensitivity	0.79963	0.68533	0.72046	0.76095	0.72777	0.73883
AUC	0.99577	0.99726	0.99602	0.99347	0.99573	0.99565
PRC	0.94712	0.94436	0.94513	0.93239	0.94285	0.94237
specificity	0.99991	0.99999	0.99998	0.99996	0.99998	0.99997
f1-score	0.87417	0.81124	0.83486	0.85832	0.83978	0.84367
Total transactions	554082.0	554081.0	554082.0	554082.0	554082.0	

Each full set is distributed as follows.

- Total number of transactions: 2770409 (100% of the dataset).
- Frauds (class 1): 8213 (100% of the total number of fraudulent transactions).
- Nonfrauds (class 0): 2762200 (100% of the total number of legitimate transactions).
- Each full set is split in *Training Set*, *Validation Set* and *Test Set*.
  - The training set has 60% of the full dataset, with 4926 fraudulent transactions and 1657320 legitimate transactions.
  - The validation set has 20% of the full dataset, with 1642 fraudulent transactions and 552440 legitimate transactions.
  - The test set has 20% of the full dataset, with 1642 fraudulent transactions and 552440 legitimate transactions.

4.3.1 Full Sets Analysis (Average)

- True Positives (TP) = 1214: These instances represent the fraudulent transactions that the model correctly identified as fraudulent. For a financial institution, these 1214 instances are situations where potential financial losses

were prevented. The aim should be to increase the number of TPs as much as possible, thereby maximizing the prevention of fraudulent activities.

- False Positives (FP) = 19: These instances represent legitimate transactions that were incorrectly flagged as fraudulent by the model. From a customer service perspective, these 19 instances could have caused unnecessary inconveniences for customers, potentially leading to customer dissatisfaction. Additionally, the institution might have spent unnecessary resources investigating these transactions, leading to operational inefficiencies. Therefore, the institution should strive to minimize the number of FPs to maintain customer satisfaction and operational efficiency.

- False Negatives (FN) = 429: These instances represent the fraudulent transactions that the model failed to detect and incorrectly labeled as legitimate. For the financial institution, these 429 instances represent missed opportunities for preventing fraud, leading to potential financial losses. Furthermore, these instances pose a risk to the institution's reputation and might potentially invite regulatory penalties. Therefore, reducing the number of FNs should be a priority to prevent potential financial and reputational damage.

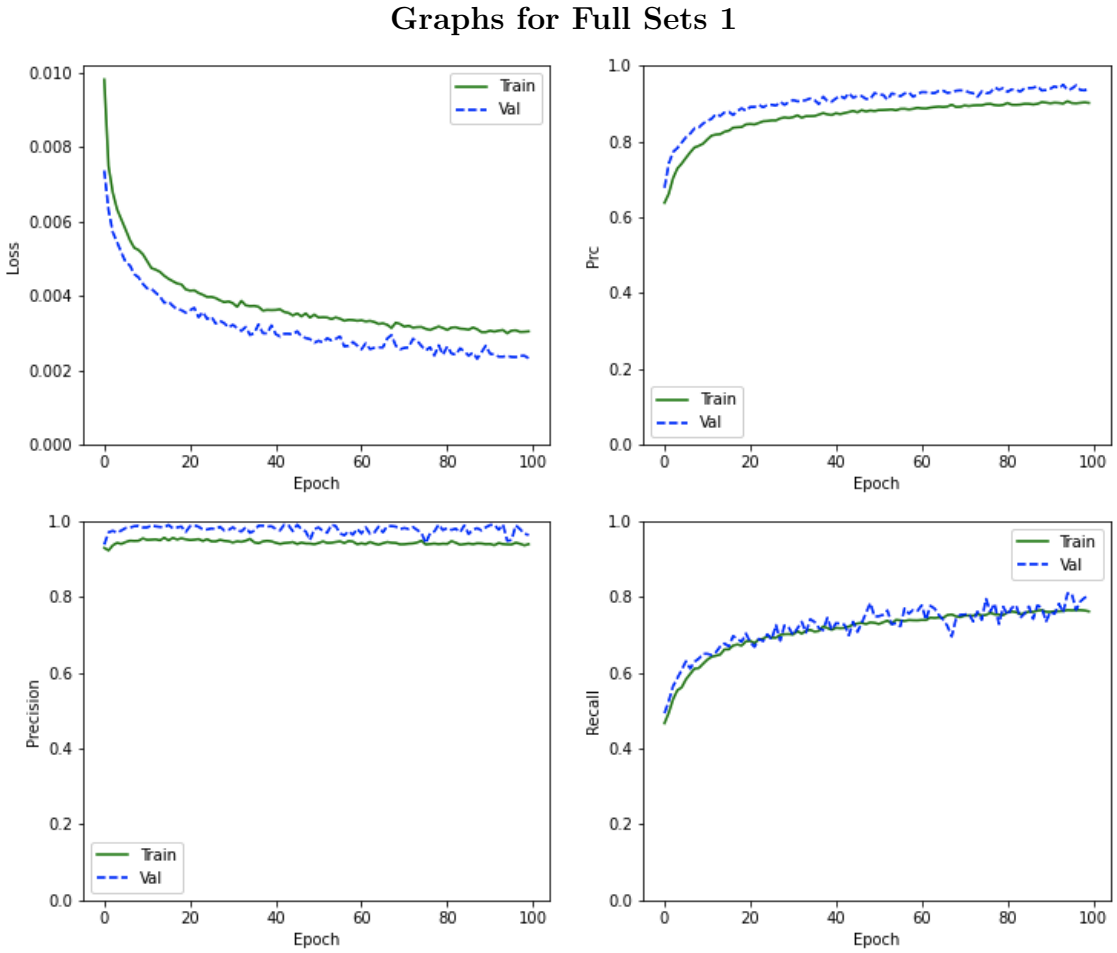

Figure 4.8: Graphs: Loss vs Epoch, PRC vs Epoch, Precision vs Epoch, and Recall vs Epoch for Full Sets 1

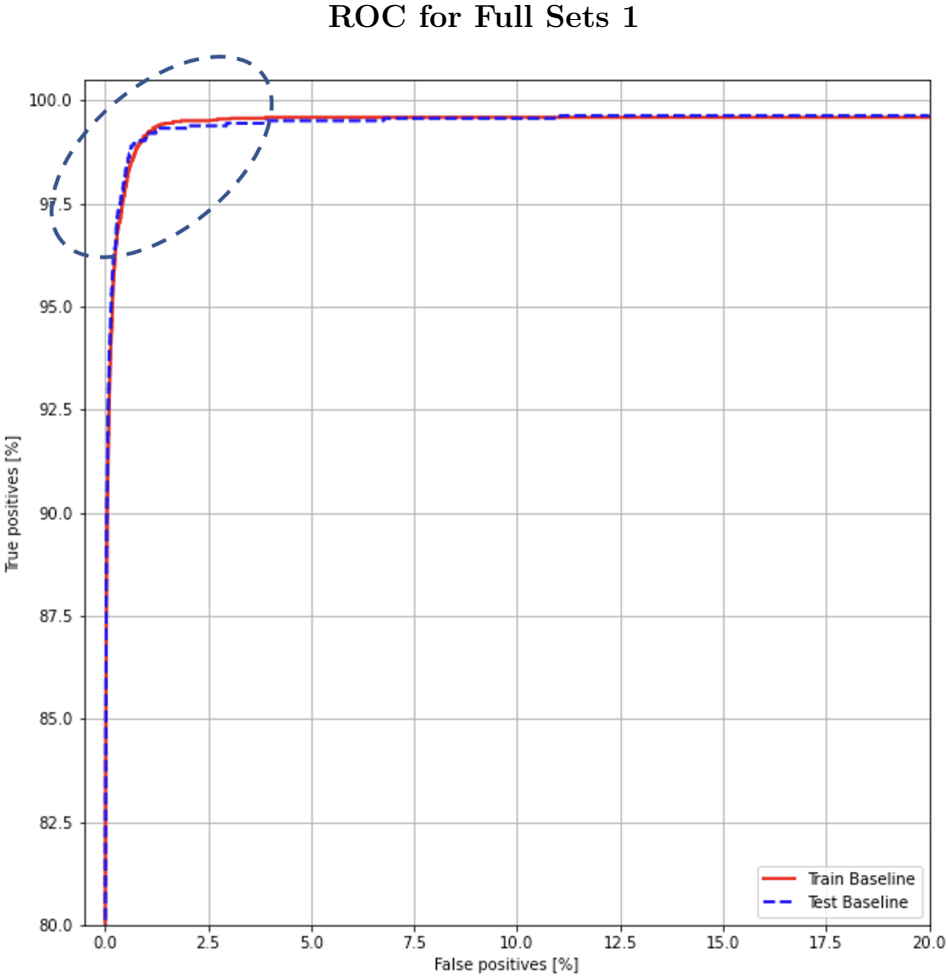

Figure 4.9: ROC for Full Sets 1

4.3.2 Full Sets average: analysis

Based on the results of the Scores, the model shows excellent performance, although there's still room for improvement. Here's a detailed analysis:

- Loss: The loss value is very low (0.00258), indicating the model's predictions are generally quite close to the true values. This suggests that the model has a very good performance in terms of overall error rates.
- Accuracy: It is excellent (0.99919), suggesting that it correctly classifies transactions as fraudulent or non-fraudulent about 99.92% of the time.
- Precision: The precision score (0.98542) is excellent. This means that when the model predicts a transaction as fraudulent, it's correct about 98.54% of the time.
- Recall (Sensitivity): The recall score is moderate at 0.73883, implying that your model identifies about 73.88% of all actual fraudulent transactions. This means the model is failing to catch a significant proportion of fraudulent transactions. In the context of fraud detection, increasing the recall is generally desirable, even if it results in more false positives.
- AUC: The AUC score is excellent (0.99565), indicating that the model does an excellent job at distinguishing between fraudulent and non-fraudulent transactions.
- PRC: This score is very good (0.94237). PRC score is often considered in cases where the dataset is unbalanced (like in fraud detection, where fraudulent transactions are far less common than non-fraudulent ones). A high score suggests that the model maintains a very good balance between precision and recall.
- Specificity: The specificity score is excellent (0.99997), which means that the model is extremely good at correctly identifying non-fraudulent transactions.
- F1-score: The F1-score (0.84367) is used as a single metric that combines precision and recall. This score is moderate, the gap between this and the precision score shows the discrepancy between precision and recall. This indicates there's room for improvement in increasing the recall.

In summary, the model demonstrates high accuracy and is excellent at predicting non-fraudulent transactions (high specificity). It is also reliable when it identifies a transaction as fraudulent (high precision). However, it fails to detect a considerable number of actual fraudulent cases (lower recall). Depending on the consequences of missing a fraud case versus the cost of investigating a false alarm, the model needs to increase recall, even if it may decrease precision.

4.3.3 Graph Analysis: Full Sets approach

There are five graphs for each model (Full Sets 1 to 5) application, they are the same as the Iterations approach graphs, as follows: Loss vs Epoch, PRC vs Epoch, Precision vs Epoch, Recall vs Epoch, and True positives [%] vs False positives [%]. For the first 4 graphs, the curves are *Train* and *Validation*, and for the last graph, the curves are *Train* and *Test*.

The qualitative analysis for these graphs are alike comparing to the ones given for the Iterations graphs, meaning that the behaviour of the curves follow similar patterns. However, there are some differences in the quantitative aspects, that indicates a better performance of the model when used in the Full Sets performances. See the insights of these analysis in the end of this chapter.

4.4 Results: Machine Learning approaches

In this section, the results of the machine learning models applied to the dataset are presented, and an evaluation is conducted to determine which method exhibits the best performance among the three. For these particular types of machine learning algorithms, the dataset applied is the one immediate following the preprocessing process, without the format technique used in the Iterations sets and in the Full Sets approach.

Table 4.3 displays the results of the scores.

Table 4.3: Scores for Random Forest, Logistic Regression and AdaBoost ML approaches

Scores	Random Forest	Logistic Regr.	AdaBoost
true positives	**1331.0**	727.0	983.0
false positives	**22.0**	746.0	50.0
true negatives	**552414.0**	551690.0	552386.0
false negatives	**315.0**	919.0	663.0
accuracy	**0.99939**	0.99700	0.99871
precision	**0.98374**	0.49355	0.95160
recall or sensitivity	**0.80863**	0.44168	0.59721
AUC	**0.90429**	0.72016	0.79856
PRC	**0.79605**	0.21965	0.56950
specificity	**0.99996**	0.99865	0.99991
f1 score	**0.88763**	0.46618	0.73386
Total transactions	554082	554082	554082

4.4.1 Analysis: Machine Learning approaches

Examining the scores of *Random Forest*, *Logistic Regression*, and *AdaBoost* from Table 4.3, we can see that Random Forest outperforms the other two models in all the scores. Specifically, Random Forest has the highest accuracy (0.99939), precision (0.98374), recall (0.80863), AUC (0.90429), PRC (0.79605), specificity (0.99996), and f1-score (0.88763) among the three models.

On the other hand, Logistic Regression has the lowest scores among the three models, with the lowest accuracy (0.99700), precision (0.49355), recall (0.44168), AUC (0.72016), PRC (0.21965), and f1-score (0.46618).

AdaBoost has relatively good performance, but not as good as Random Forest. AdaBoost has a higher precision (0.95160) and f1-score (0.73386) than Logistic Regression, but still lower than Random Forest. However, AdaBoost has a better recall (0.59721) and AUC (0.79856) than Logistic Regression, although still lower than Random Forest.

As a conclusion, Random Forest performs better than AdaBoost and Logistic Regression. AdaBoost performs better than Logistic Regression.

4.5 Analysis: Iterations vs Full Sets

In this section, the analytical comparison of the performances between the Iterations scores and the Full Sets scores is performed based on the their averages, to determine what is the best dataset format distribution between them to apply the neural network model. To represent the overall performance of the model, I calculate the arithmetic average of these metric scores across all five folds from Iterations and Full Sets approaches. This kind of data splitting mitigates the risk of an overly optimistic or pessimistic estimate of the model's performance due to a particularly favourable or unfavourable split of data. The average metric scores provide a more robust and reliable estimate of the model's performance on unseen data.

Table 4.4 displays the results.

Table 4.4: Scores for Iterations and Full Sets approaches (averages)

Scores	Aver. Iterations	Aver. Full Sets
loss	0.00371	**0.00258**
accuracy	0.99909	**0.99919**
precision	0.98287	**0.98542**
recall	0.70222	**0.73883**
AUC	0.98416	**0.99565**
PRC	0.88761	**0.94237**
specificity	0.99996	**0.99997**
f1 score	0.81876	**0.84367**

- **Analysis:**

Comparing the two sets of results from the Iterations and the Full Sets approaches, Full Sets approach has better results in all evaluation scores than the Iterations approach.

In terms of accuracy, the Full Sets (99.92%) shows a similar result comparing to the Iterations (99.91%). The difference is minimal, and statistically insignificant.

In precision, the Full Sets results (98.54%) also shows similar performance comparing to the Iterations result (98.28%).

Recall, which measures the number of true positive cases divided by the total number of actual positive cases, has shown an improvement in the Full Sets results

(73.88%) compared to the Iterations results (70.22%), however, it is not considered statistically significant.

The Area Under the Curve (AUC) metric is used to evaluate the overall performance of the model. Full Sets approach has again similar results (99.56%) comparing to the Iterations result (98.41%).

The Precision-Recall Curve (PRC) measures the tradeoff between precision and recall, and the Full Sets results (94.23%) shows an improvement over the Iterations results (88.76%), but still not statistically significant at all.

The specificity score is the same for both approaches, 99.99%.

In financial fraud detection, the most important scores are typically Recall (also known as Sensitivity or True Positive Rate) and Precision. These two measures are particularly relevant since financial institutions often need to balance the cost of false alarms (which can be high if customer transactions are unnecessarily blocked, leading to customer dissatisfaction and potential loss of business) against the cost of missing a fraudulent transaction.

Recall is crucial because a high recall indicates a low number of false negatives, meaning fewer fraudulent transactions are missed. Meanwhile, precision is also important as a high precision means a low number of false positives, indicating that when the model predicts a transaction is fraudulent, it is likely to be correct.

Morgan Grandi [45] in 2020 published a blog discussing about the trade-offs between false positives (legitimate transactions predicted as fraudulent) against false negatives (fraudulent transactions predicted as legitimate) in the perspective of the financial institutions.

As a conclusion, the Full Sets performance is better than the Iterations performance in all aspects based on the scores results.

4.5.1 ROC Curves: Performances

In a ROC curve, the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of

how well a parameter can distinguish between two classes (1 or 0).

A perfect classifier would have a ROC curve that passes through the top left corner, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). This way, the closer the ROC curve is to the top left corner, the higher the overall accuracy of the test.

An important aspect of the ROC curve is that it shows the trade-off between sensitivity and specificity. Any increase in sensitivity will usually be accompanied by a decrease in specificity. The best test would be one that increases the proportion of true positive results while minimizing false positive results.

The AUC represents a measure of separability or how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. An excellent model has AUC near to the 1, which means it has a good measure of separability. A poor model has AUC near to the 0, which means it has the worst measure of separability, in fact it is reciprocating the result. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

The diagonal line from the bottom left to the top right is the line of no-discrimination, which refers to a test that cannot discriminate between the diseased and the healthy. A classifier that performs no better than random chance will have an ROC curve that lies along this diagonal.

Examining the ROC graphs in Figures 4.10 and 4.11, it can be concluded that the plots number 2, 4, 6, 8 and 10 from the Full Sets approach shows a better performance than what is showing in 1, 3, 5, 7, and 9 from the Iterations approach. Besides the curves being closer to the top left corner of the plot on Full Sets graphs (train baseline and test baseline curves), they flow steadier than the curves in graphs from Iterations approach.

As a conclusion, although the Full Sets approach presented slightly better curves, they are considered statistically similar, and both approaches demonstrated unexpected similar performances.

In security contexts, such as intrusion detection or fraud detection, the datasets are usually highly unbalanced, with a large number of non-fraudulent (negative) instances and a small number of fraudulent (positive) instances. This skew in the class distribution can make ROC curves misleading because the ROC curve might show a high area under curve (AUC) value, implying good performance, even if the classifier is performing poorly in correctly identifying the minority (positive) class, which is

often the class of primary interest in security contexts. This is because the ROC curve plots True Positive Rate (TPR) against False Positive Rate (FPR), and with a highly unbalanced dataset, even a simple model that always predicts the majority class will achieve a high TPR and low FPR, leading to a seemingly good ROC curve.

For these reasons, other measurements should be considered as well. ROC curves are primarily useful in binary classification problems where both classes (positive and negative) are more or less balanced.

ROC Curve graphs

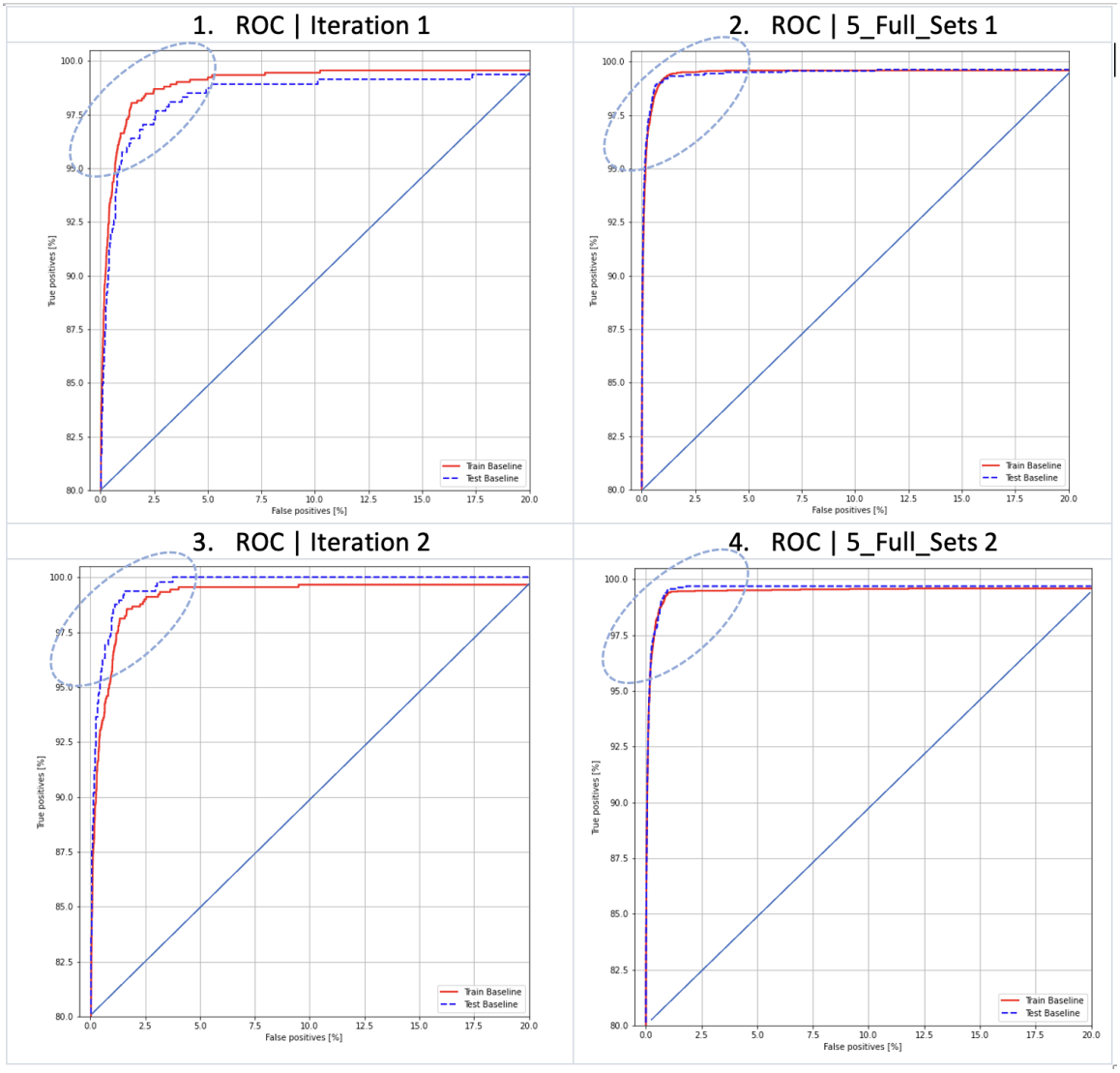

Figure 4.10: ROC graphs highlighting the curves for performance comparison (Iterations vs Full Sets, 1 and 2)

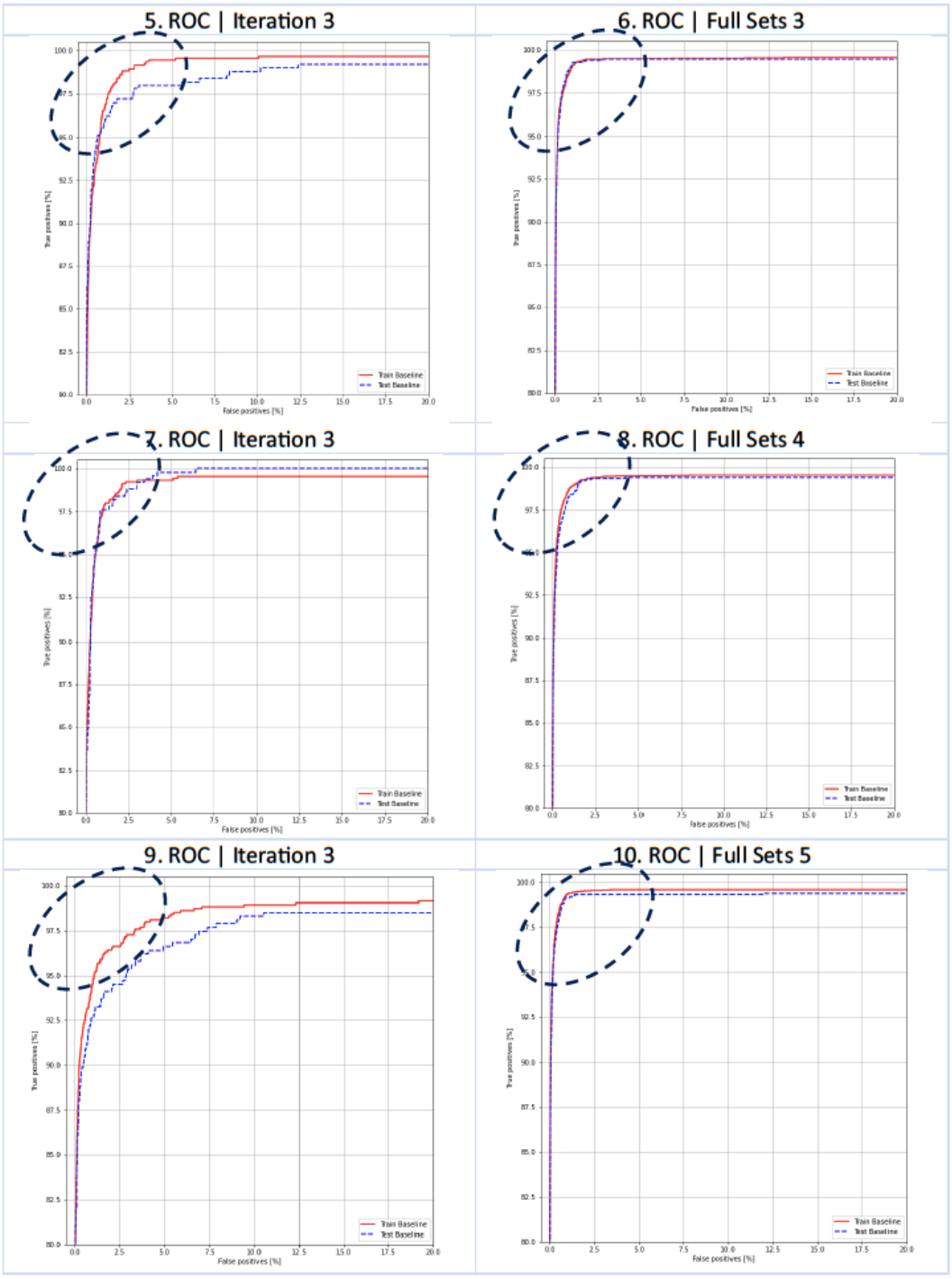

Figure 4.11: ROC graphs highlighting the curves for performance comparison (Iterations vs Full Sets, 3, 4, and 5)

4.6 Analysis: Random Forest vs Full Sets

In this section, I present the analysis and comparison of the performances between the Random Forest and five Full Sets approaches (Table 4.5).

Table 4.5: Scores from Random Forest and Full Sets average

Scores	Random Forest	Aver. Full Sets
accuracy	**0.99939**	0.99919
precision	0.98374	**0.98542**
recall	**0.80863**	0.73883
AUC	0.90429	**0.99565**
PRC	0.79605	**0.94237**
specificity	0.99996	**0.99997**
f1 score	**0.88763**	0.84367

- **Analysis:**

Based on the score results, the indication is that the Random Forest algorithm performs better in two scores, recall or sensitivity (0.80863 vs 0.73883) and f1 score (0.88763 vs 0.73883); the Full Sets approach has better performance in two scores, AUC (0.99565 vs 0.90429) and PRC (0.94237 vs 0.79605), and they have similar performances in accuracy (0.99939 vs 0.99919), precision (0.98542 vs 0.98374), and specificity (0.99997 vs 0.99996).

Highlights:

- Random Forest performs better than the Full Sets approach: Recall (0.80863 vs 0.73883) and f1 score (0.88763 vs 0.84367), meaning that the model is more effective at identifying fraudulent transactions (it has higher recall) and has a better balance between identifying positive cases and avoiding false positives (it has a higher F1 score).
- Full Sets performs better than Random Forest: AUC (0.99565 vs 0.90429) and PRC (0.94237 vs 0.79605), meaning that the Full Sets model, having a higher AUC, is better at distinguishing between fraudulent and legitimate transactions compared to the Random Forest model, and with a higher PRC, the Full Sets

model shows a better balance between the ability to detect fraudulent transactions (recall) and the ability to avoid false alarms (precision).

This thesis proposes a modified form of k-fold cross-validation in conjunction with a neural network algorithm applied to the PaySim dataset. This technique generates five distinct subsets from the same dataset, ensuring that 100% of the data is used across all sets. This approach has superior performance compared to the machine learning method, the Random Forest algorithm applied to the preprocessed dataset without any special formatting, using the dataset as a whole just once.

The next part of this thesis is the Discussions, Chapter 5, presenting the findings and discussions of this thesis.

Chapter 5

Discussions

In this chapter, I present the analysis and interpretation of the results obtained from this thesis, including a summary of the key findings, an interpretation of their meaning, a comparison to existing research, an identification of limitations and implications, and a conclusion. In summarizing the results, I highlight the most important findings and their significance. During the interpretation section I explain how my results relate to the research questions and hypotheses, and what insights they offer. I also compare my results to existing research and show the contributions my research makes to the field. I acknowledge and consider the limitations and implications of this thesis, so I summarize the key points and highlight the overall significance of my research to the field.

5.1 Results Summary

The first evaluation was to calculate the average of the scores of the Iterations approach, after applying the neural network model in all five iterations.

Each set is represented as follows:

- 20% of the full dataset;
- with 1642 class 1 (frauds) samples and 552440 class 0 (nonfrauds) samples.

The second evaluation was to calculate the average of the scores of the Full Sets approach, after applying the neural network model in all five sets.

Each set is represented as follows:

- 100% of the full dataset;
- with 8213 class 1 (frauds) samples and 2762200 class 0 (nonfrauds) samples.

The third evaluation was to compare the performance between the results of the first evaluation, Iterations average, and the second evaluation, Full Sets average, and determine the best performance.

The fourth evaluation was to compare the performance among the three machine learning algorithms Random Forest, Logistic Regression and AdaBoost and determine the best method, based on the scores, after utilizing each of the them in the classification process of the full PaySim dataset.

The fifth evaluation was to compare the winner method of the third evaluation (Iterations vs Full Sets) to the winner of the fourth evaluation (machine learning methods) and determine the best performance. Table 5.1 shows the summary of the five evaluations.

Table 5.1: Evaluations summary

Evaluations	**Description**
1^{st}	Iterations average
2^{nd}	Full Sets average
3^{rd}	Iterations vs Full Sets
4^{th}	Machine Learning approaches
5^{th}	Machine Learning vs (Iterations or Full Sets)

5.1.1 Results: Iterations approach

Summary considering the averages of the five iterations, Table 5.2:

Table 5.2: Results of the five iterations (each with 20% of dataset) measurements and the average

Scores	Iterations Average
loss	0.00371
true positives	**342**
false positives	**6**
true negatives	165732
false negatives	**145**
accuracy	0.99909
precision	0.98287
recall or sensitivity	0.70222
AUC	0.98416
PRC	0.88761
specificity	0.99996
f1 score	0.81876
Total transactions	166225.0

- Loss: 0.00371, indicating this value as the difference between the predicted probability distribution and the true probability distribution.
- True Positives (TP): 342, False Positives (FP): 6, True Negatives (TN): 165732, False Negatives (FN): 145.
- Accuracy: It is the percentage of correctly classified examples out of all the examples in the test set, 0.99909 (99.9%).
- Precision: The precision value means the percentage that the model predicts an example to be positive, and it is likely to be correct. In this case, the value is 0.98287 (98.3%).
- Recall (sensitivity): The proportion of true positive predictions (predicted as frauds) out of all the actual positive examples (all fraudulent transactions), and in this case, the recall is 0.70222 (70.2%).
- AUC: It ranges between 0 and 1, where a value of 1 indicates perfect performance and a value of 0.5 indicates random performance (explained in detail in Chapter 2). In this case, the AUC is 0.98416 (98.4%).

- PRC: The area under the precision-recall curve is another popular metric for evaluating binary classification models. It ranges between 0 and 1, where a value of 1 indicates perfect performance and a value of 0 indicates poor performance. In this case, the PRC is 0.88761 (88.7%).
- Specificity: The proportion of true negative predictions out of all the actual negative examples in the test set. In this case, the specificity is 0.99996 (99.99%).
- F1 score: It is a measure of a model's accuracy that combines precision and recall into a single metric (trade-off between precision and recall). A high F1 score indicates that a model has both high precision and high recall, meaning that it is good at both identifying true positive cases (fraudulent transactions) and avoiding false positive and false negative predictions. In this case, the F1 score is 0.81876 (81.9%).

5.1.2 Results: Full Sets approach

Summary considering the averages of the five sets, Table 5.3:

Table 5.3: Full Sets average

Scores	**Full Sets Average**
loss	0.00258
true positives	**1214**
false positives	**19**
true negatives	552420
false negatives	**429**
accuracy	0.99919
precision	0.98542
recall or sensitivity	0.73883
AUC	0.99565
PRC	0.94237
specificity	0.99997
f1-score	0.84367
Total transactions	554082

- Loss: 0.00258.
- True Positives (TP): 1214, False Positives (FP): 19, True Negatives (TN): 552420, False Negatives (FN): 429.
- Accuracy: The percentage of correctly classified examples out of all the examples in the test set. The accuracy of the model is 0.99919 (99.92%).
- Precision: The precision value is the percentage the model predicts an example to be positive, and it is likely to be correct. In this case, the precision is 0.98542, indicating that when the model predicts a positive instance, it is correct 98.5% of the time.
- Recall (sensitivity): The recall is 0.73883, indicating that the model has 73.9% of probability of identifying correctly the positive examples.
- AUC: The AUC is 0.99565, indicating that the model has 99.6% of discriminatory power rate.
- PRC: The PRC score is 0.94237 (94.3%).
- Specificity: A high specificity value means that the model is good at identifying negative examples, and in this case, the specificity is 0.99997 (99.9%).
- F1 score: The F1 score is 0.84367, which is the balance between precision and recall.

5.1.3 Results: Machine Learning approaches

From Table 5.4, after analysis, the conclusion is that the Random Forest (RF) algorithm has the best performance comparing to the other two machine learning models Logistic Regression (LR) and AdaBoost (AB), in all scores.

- RF has the highest scores in accuracy (0.99939), precision (0.98374), recall (0.80863), AUC (0.90429), PRC (0.79605), specificity (0.99996), and f1-score (0.88763).

- LR has the lowest scores among the three models, with the lowest accuracy (0.99700), precision (0.49355), recall (0.44168), AUC (0.72016), PRC (0.21965), and f1-score (0.46618).
- Although AB has relatively good performance, it is not as good as RF.
- AB has a higher precision (0.95160) and f1-score (0.73386) than LR but lower than RF.
- AB has a better recall (0.59721) and AUC (0.79856) than LR, although still lower than RF.
- In conclusion, RF is the best-performing model, followed by AB and then LR.

5.1.4 Results: Iterations vs Full Sets

Comparing the two sets of results from the Iterations and the Full Sets approaches, the conclusion is that the Full Sets approach has better performance than the Iterations approach, in all scores. See Table 5.5.

5.1.5 Results: Random Forest vs Full Sets

The results (see Table 5.4) shows:

- Random Forest algorithm performs better in three scores:
  - in accuracy (0.99939 vs 0.99919)
  - in recall or sensitivity (0.80863 vs 0.73883)
  - in f1 score (0.88763 vs 0.84367)

- Full Sets approach has better performance in four scores:

  - in precision (0.98542 vs 0.98374)
  - in AUC (0.99565 vs 0.90429)
  - in PRC (0.94237 vs 0.79605)
  - in specificity (0.99997 vs 0.99996)

To summarize,

- Random Forest method is better at correctly classifying both fraud and legitimate transactions. It also identifies more fraud transactions correctly. And it balances the precision and recall more effectively.
- Full Sets approach is better at distinguishing between frauds and legitimate transactions. It is also better at achieving both high precision and high recall. It is better at accurately identifying frauds. And it is better at identifying legitimate transactions.
- The Random Forest approach has better performances in three scores, against three scores in which the Full Sets method performs better.
- The scores in which the Full Sets approach does not perform better than the Random Forest method, the difference is considerably insignificant, while in 2 scores out of the 4 that the Full Sets approach performs better, the difference is relevant, and they are AUC (0.99565 vs 0.90429) and PRC (0.94237 vs 0.79605).

Table 5.4: All three machine learning approaches and the Full Sets scores

Scores	**RF**	**LR**	**AB**	**Aver. FS**
true positives	1331.0	727.0	983.0	1214
false positives	22.0	746.0	50.0	19
true negatives	552414.0	551690.0	552386.0	552420
false negatives	315.0	919.0	663.0	429
accuracy	0.99939	0.99700	0.99871	0.99919
precision	0.98374	0.49355	0.95160	0.98542
recall or sensitivity	0.80863	0.44168	0.59721	0.73883
AUC	0.90429	0.72016	0.79856	0.99565
PRC	0.79605	0.21965	0.56950	0.94237
specificity	0.99996	0.99865	0.99991	0.99997
f1-score	0.88763	0.46618	0.73386	0.84367

Table 5.5: All the Iterations and Full Sets with their respective averages

Scores	**Iter. 1**	**Iter. 2**	**Iter. 3**	**Iter. 4**	**Iter. 5**	**Aver.**	**FS 1**	**FS 2**	**FS 3**	**FS 4**	**FS 5**	**Aver.**
loss	0.00334	0.00300	0.00405	0.00341	0.00478	0.00371	0.00217	0.00279	0.00262	0.00264	0.00269	0.00258
true positives	349.0	350.0	354.0	349.0	308.0	342.0	1313.0	1126.0	1183.0	1251.0	1195.0	1213.6
false positives	7.0	7.0	4.0	8.0	4.0	6.0	49.0	7.0	9.0	20.0	9.0	18.8
true negatives	165746.0	165729.0	165725.0	165713.0	165747.0	165732.0	552391.0	552431.0	552431.0	552418.0	552431.0	552420.4
false negatives	123.0	139.0	142.0	155.0	166.0	145.0	329.0	517.0	459.0	393.0	447.0	429.0
accuracy	0.99922	0.99912	0.99912	0.99902	0.99898	0.99909	0.99932	0.99905	0.99916	0.99925	0.99918	0.99919
precision	0.98034	0.98039	0.98883	0.97759	0.98718	0.98287	0.96402	0.99382	0.99245	0.98426	0.99252	0.98542
recall or sensitivity	0.73941	0.71575	0.71371	0.69246	0.64979	0.70222	0.79963	0.68533	0.72046	0.76095	0.72777	0.73883
AUC	0.98099	0.99624	0.98498	0.99120	0.96737	0.98416	0.99577	0.99726	0.99602	0.99347	0.99573	0.99565
PRC	0.87847	0.90886	0.90128	0.88562	0.86381	0.88761	0.94712	0.94436	0.94513	0.93239	0.94285	0.94237
specificity	0.99996	0.99996	0.99998	0.99995	0.99998	0.99996	0.99991	0.99999	0.99998	0.99996	0.99998	0.99997
f1-score	0.84299	0.82742	0.82903	0.81068	0.78370	0.81876	0.87417	0.81124	0.83486	0.85832	0.83978	0.84367

5.2 Findings

5.2.1 False Negatives vs False Positives

The Trade-Offs

In financial fraud detection, as with many other binary classification problems, there is often a trade-off between false positives (FP) and false negatives (FN).

A false positive in this context is when the system incorrectly predicts a transaction as fraudulent when it is actually legitimate. This might cause some inconvenience to the customer who has to prove their identity or validate the transaction, leading to a negative customer experience. It also consumes resources of the fraud team who have to manually review these false alarms. It happens mainly because the base rate fallacy is not taken into account when analysing a classification problem like this.

A false negative is when the system fails to detect an actual fraudulent transaction. This could potentially cause substantial financial losses if the fraudulent transactions are of high value. This might also hurt the company's reputation and trustworthiness if such issues become known to the public.

In the ideal world, we would want to minimize both FP and FN. However, in practice, reducing one often affects the other to increase. For example, if we tighten the rules or thresholds to reduce FN, we might end up flagging more legitimate transactions as fraudulent, increasing the FP rate. If we loosen the rules to reduce FP, we might end up missing more actual fraud cases, causing a higher FN rate.

Finding the best trade-off between FP and FN depends on the specific needs and risk tolerance of the financial institution. For instance, a company that values customer experience highly might be willing to tolerate a higher FN rate in exchange for a lower FP rate. On the other hand, a company that operates in a high-risk market or has a smaller margin for loss might prioritize reducing FN even if it means a higher FP rate.

This balance can be adjusted by changing the threshold for classification, employing different machine learning models, using specific techniques, constrained fundamentally by the Bayes error rate [46].

In any case, it is important to remember that the ultimate goal is not just to op-

timize a certain metric or achieve the lowest possible FP or FN rate, but to effectively manage and minimize the overall risk and impact to the business. This requires a deep understanding of the business context, the cost implications of FP and FN, as well as the capabilities and limitations of the machine learning models employed.

The Threshold

Determining the threshold for a specific false positive and false negative rate from a precision-recall curve is described as follows.

Firstly, it's important to note that precision (also known as positive predictive value) is the fraction of true positives among the total predicted positives (true positives + false positives), while recall (also known as sensitivity) is the fraction of true positives among the total actual positives (true positives + false negatives).

Precision and recall are inversely related. As the threshold decreases (making the classifier more liberal), recall increases and precision decreases. On the other hand, as the threshold increases (making the classifier more conservative), precision increases and recall decreases. The precision-recall curve shows this trade-off for different thresholds.

With a precision-recall curve for the model, and having the number of actual positive cases, it is possible to infer the false positive and false negative rates indirectly through precision and recall.

False Negatives: If Recall = TP/(TP+FN), then FN = TP*(1/Recall - 1). The true positives (TP) are the actual positives multiplied by recall: TP = Actual Positives * Recall.

False Positives: You can calculate the number of false positives directly from precision and the number of true positives. If Precision = TP/(TP+FP), then FP = TP*(1/Precision - 1).

This approach will give the number of false positives and false negatives for every point on the precision-recall curve, or every possible threshold.

Once the false positive and false negative rates of interest are determined, the point on the curve that corresponds to these rates, is the optimal threshold.

The precision-recall curve only gives a general sense of how the false positive and false negative rates trade-off for different thresholds. It does not calculate the exact threshold values. For this, it is possible to directly inspect the predicted probabilities

generated by the model and calculate the rates for different threshold values manually.

In practice, machine learning libraries like scikit-learn provide functions to calculate the threshold directly:

```
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt

# Assuming y_test are your true binary labels and y_score are
# the predicted probabilities
precisions, recalls, thresholds = precision_recall_curve(y_test, y_score)

# Plot Precision-Recall vs Threshold curve
plt.figure(figsize=(8, 6))
plt.plot(thresholds, precisions[:-1], 'b--', label='Precision')
plt.plot(thresholds, recalls[:-1], 'g-', label='Recall')
plt.xlabel('Threshold')
plt.legend(loc='best')
plt.grid()
plt.title('Precision-Recall vs Threshold')
plt.show()
```

The precision_recall_curve function returns three arrays:

- Precisions: Precision values such that element i is the precision of predictions with score $\geq$ thresholds[i] and the last element is 1.
- Recalls: Decreasing recall values such that element i is the recall of predictions with score $\geq$ thresholds[i] and the last element is 0.
- Thresholds: Increasing thresholds on the decision function used to compute precision and recall.

By plotting precisions and recalls over thresholds, it is possible to observe how precision and recall change with different threshold values. This allows to select a threshold that gives the best trade-off between precision and recall for specific problem of interest.

A high precision means a low false positive rate, and a high recall means a low false negative rate. Selecting the right trade-off depends on the cost of false positives vs. false negatives in each specific use-case, and the Bayes error rate is the optimal decision boundary that minimizes risk.

On the other hand, reducing false positives in fraud detection systems is not just a matter of improving the accuracy of the system, but is crucial to maintaining customer satisfaction, ensuring operational efficiency, protecting reputation, and ultimately, the survival and growth of the business.

Addressing fraudulent activity is undoubtedly a critical task within any business; however, this responsibility can occasionally lead to an inadvertent bias among fraud examiners, who may place disproportionate emphasis on this aspect alone. Consequently, this could result in the overlooking of other equally vital facets of the business. This is a delicate balancing act that organizations must manage carefully, ensuring that while fraud prevention remains a priority, it does not eclipse attention from other essential operational areas.

False Negatives & False Positives of Interest

In this section, I present a technique to manage the ratio *false positives/false negatives*, with the goal to find an optimal threshold between them. I use the graph (Figure 5.1) Precision vs Recall with Train and Test baseline curves, from the Full Sets 1. I used an online graph reader tool called Graphreader [47], to plot the points A, B, C, D, E, F, G, H, I , J, K, L, and M on the graph. The coordinates of the points are represented by Precision (x-axis) and Recall (y-axis). I used three points: G(0.695, 0.940), I(0.800, 0.915), and K(0.895, 0.875).

1. Point G: Precision = 0.695, Recall = 0.940, Actual Positives = 1642

$$\text{True Positives (TP)} = \text{Recall} \times \text{Actual Positives} = 0.94 \times 1642 = 1544$$
$$\text{False Negatives (FN)} = \text{Actual Positives} - \text{TP} = 1642 - 1544 = 98$$
$$\text{False Positives (FP)} = \frac{\text{TP}}{\text{Precision}} - \text{TP} = \frac{1544}{0.695} - 1544 = 666$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.695 \times 0.94}{0.695 + 0.94} = 0.798$$

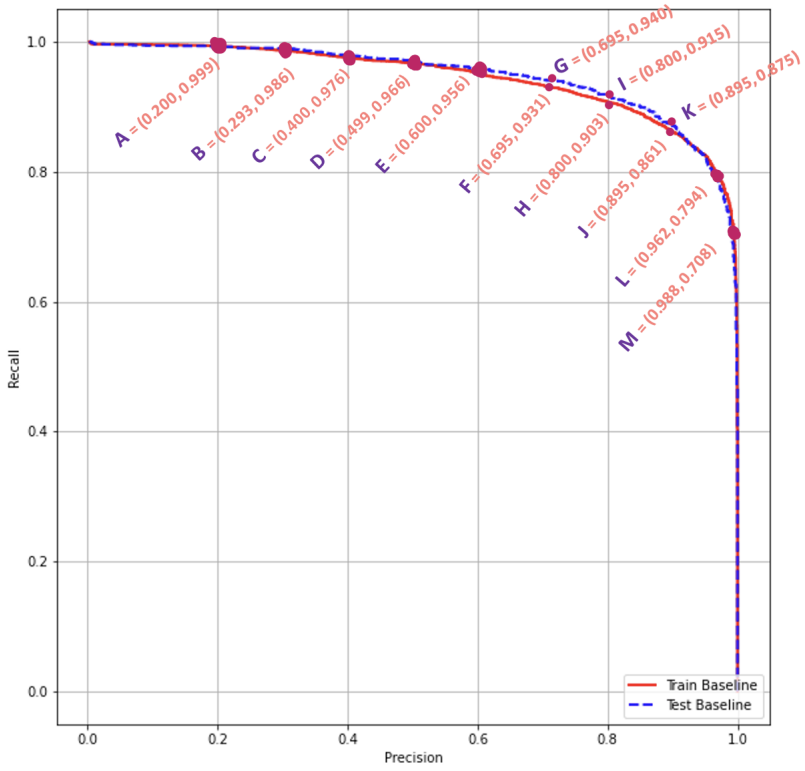

Figure 5.1: Precision vs Recall graph with Train and Test curves for Full Set 1

2. Point I: Precision = 0.800, Recall = 0.915, Actual Positives = 1642

$$\text{True Positives (TP)} = \text{Recall} \times \text{Actual Positives} = 0.915 \times 1642 = 1502$$

$$\text{False Negatives (FN)} = \text{Actual Positives} - \text{TP} = 1642 - 1502 = 140$$

$$\text{False Positives (FP)} = \frac{\text{TP}}{\text{Precision}} - \text{TP} = \frac{1502}{0.8} - 1502 = 375$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.8 \times 0.915}{0.8 + 0.915} = 0.854$$

3. Point K: Precision = 0.895, Recall = 0.875, Actual Positives = 1642

$$\text{True Positives (TP)} = \text{Recall} \times \text{Actual Positives} = 0.875 \times 1642 = 1437$$
$$\text{False Negatives (FN)} = \text{Actual Positives} - \text{TP} = 1642 - 1437 = 205$$
$$\text{False Positives (FP)} = \frac{\text{TP}}{\text{Precision}} - \text{TP} = \frac{1437}{0.895} - 1437 = 170$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.895 \times 0.875}{0.895 + 0.875} = 0.885$$

In summary (Table 5.6), these results demonstrate the trade-off between precision and recall in the model's performance. Depending on the specific needs and tolerance for false positives and false negatives, a financial institution can choose a model configuration that best suits their requirements. Point K, for example, might be preferred if the goal is to minimize false positives, while point G could be chosen if the highest recall is desired.

Table 5.6: Summary of the performance at points G, I, and K

Point	Precision	Recall	False Negatives	False Positives	F1-score
G	0.695	0.940	98	666	0.798
I	0.800	0.915	140	375	0.854
K	0.895	0.875	205	170	0.885

5.2.2 Addressing the Research Questions

- **RQ1: Which approach has the best results, the Iterations method that used 5 sets of 20% of dataset or the Full Sets approach that used sets of 100% of dataset applied to the same neural network model?**

  The Full Sets approach typically leads to better performance because the neural network is trained on the entire dataset, which provides more data for the model to learn from. In contrast, the Iterations approach involves training the model on smaller subsets of the data, which may not provide sufficient information for

the model to generalize well to new data. When using the Full Sets approach, the neural network has access to more examples of each class, which helps it learn more robust representations of the data. Additionally, by training on the entire dataset, the model can identify patterns and relationships in the data that may not be apparent in smaller subsets.

Furthermore, in the Full Sets approach, the data is divided into five sets, each one with its own unique training, testing, and validation sets. This means that the model is trained on all available data, tested on all available data, and validated in all available data. Each set is used in turns as the validation set, and test set, while the other three sets are used for training. This approach allows for a comprehensive evaluation of the model's performance and provides a more accurate estimate of its generalization ability. By using all available data for training and testing, the model can better capture the underlying patterns in the data and produce more accurate predictions.

Concluding, the **Hypothesis 1** (HP1: The approach that has better results is the Full Sets, over the Iterations and the machine learning algorithms), is confirmed and the results proved it to be consistent.

- **RQ2: Which traditional machine learning method has the best results, Random Forest, Logistic Regression or AdaBoost?**

  Random Forest, AdaBoost, and Logistic Regression are all machine learning algorithms used for binary classification problems. However, Random Forest is usually preferred over AdaBoost and Logistic Regression because of its ability to handle high-dimensional and complex data. Here are some reasons why Random Forest can outperform AdaBoost and Logistic Regression:

  - Ensemble method: Random Forest is an ensemble method that combines multiple decision trees, each built on a random subset of the features and data, to create a more robust and accurate model. This approach reduces the risk of overfitting and improves the model's generalization ability.
  - Handling nonlinear relationships: Random Forest can handle nonlinear relationships between the features and the target variable, making it well-suited for complex data. In contrast, AdaBoost and Logistic Regression

are linear models, which assume that the relationships between the features and the target variable are linear.

- Dealing with missing data: Random Forest can handle missing data by using the available information to make predictions, whereas AdaBoost and Logistic Regression require complete data to make predictions.
- Feature importance: Random Forest provides a measure of feature importance, which helps to identify the most relevant features in the dataset.
- Robustness to outliers: Random Forest is less sensitive to outliers compared to AdaBoost and Logistic Regression, which can be heavily influenced by extreme values in the data.

So the results indicates that Random Forest has better performance than Logistic Regression and AdaBoost, for this particular dataset, and it confirms the **Hypothesis 2** (HP2: Random Forest has the best performance comparing to Logistic Regression and AdaBoost), being the results consistent with it.

- **RQ3: How does the best approach submitted to the neural network model compare to the best machine learning approach in performance?**

  The results show that Random Forest performs better in three scores: accuracy, recall or sensitivity, and f1-score. On the other hand, the Full Sets approach performs better in four other scores: precision, AUC, PRC, and specificity.

  The results indicates that the slight difference in the performance scores in favor of Full Sets against Random Forest, are not statistically significant, they both perform similar.

5.2.3 True Positives, False Positives, True Negatives, False Negatives.

The research highlights the importance of identifying false positives and false negatives in financial fraud detection. While high true positive and true negative rates are desirable, the focus should be on reducing the false negative rate. This is because a

false negative can result in missed opportunities to detect fraudulent activity, which can be detrimental to the financial industry.

Therefore, reducing the false negative rate should be prioritized over reducing the false positive rate, even if it results in a higher number of false positives. This means that financial fraud detection systems should prioritize identifying as many potential fraudulent activities as possible, even if some of those activities turn out not to be fraudulent. These findings emphasize the need to improve the accuracy of fraud detection systems and minimize risks associated with fraudulent activities in the financial industry.

According to a study by the Association of Certified Fraud Examiners, reducing fraudulent transactions (false negatives) is considered the most important factor in the effectiveness of fraud detection systems. The study found that reducing false negatives was more important than minimizing false positives, which can sometimes be seen as a necessary cost of catching fraud [5].

Nevertheless, the repercussions of high false positive rates can be quite damaging. Frequent false positives can lead to dissatisfaction and may even prompt customers to switch to a different service provider, leading to a loss of business for the company. While it's critical to catch as many fraudulent transactions as possible, the necessity of reducing false positives is equally vital.

5.3 Threats to Validity

In the context of quantitative research, Campbell in [48, 49] states that **threats to validity** refer to factors that could undermine the internal or external validity of a study. Internal validity refers to the extent to which a study's results are due to the manipulation of the independent variable, rather than unrelated factors. External validity refers to the extent to which a study's results can be generalized to other populations or settings.

In the case of binary classification with highly unbalanced datasets, one classic threat to validity is **sample bias**. It occurs when the classes being classified are not represented equally in the sample, which can lead to biased models that are optimized for the majority class. This can result in poor performance on the minority class,

which may be the class of interest in many applications.

- *Addressed:* In this thesis, sample bias was addressed through the utilization of a modified form of 5-fold cross-validation in the dataset, during the evaluation of Full Sets methods. In other words, this proposed technique ensured that the five sets had classes proportionally distributed each one and they were representative of the full dataset, as well as the subsets for training, validation, and test, had all proportional class distribution.

Another potential threat to validity is **measurement bias**. In binary classification problem with unbalanced datasets, the performance of a model is typically evaluated using measurements such as accuracy, precision, recall, and F1 score. However, only using these scores can be misleading when applied to highly unbalanced datasets, as they may not accurately reflect the performance of the model on the minority class. So alternative measurements score, such as AUC (area under the curve) and ROC (receiver operating characteristic) curve, should be included in the evaluation.

- *Addressed:* This issue was handled, through utilizing a variety of alternative measurements scores. Several studies in the field considered to use a combination of scores for evaluating the performance of models on unbalanced datasets.

Confounding variables may also be a threat to validity in binary classification with unbalanced datasets. For example, the class unbalance may be related to other variables in the dataset, such as the sampling method or the presence of outliers. Failing to account for these variables can lead to biased models that are not generalizable to other datasets.

- *Not Addressed:* This issue was not directly handled in this thesis, however, there are ways to manage the effects of confounding variables. One approach is to use statistical techniques such as regression analysis [50] to control for the effects of confounding variables on the dependent variable. Another way is to conduct a sensitivity analysis [51] to explore the impact of potential confounding variables on the study's findings.

Class unbalance is another threat of validity that may impact the performance of the model, and it occurs when the number of observations in one class is much larger or smaller than the number of observations in the other class. However, in fraud detection studies, it is quite common to occur a relevant unbalance on the dataset because the frauds are the minority class. The unbalance can not be avoided in the raw dataset due to the nature of information, however there are techniques to manage it. Class unbalance can lead to poor model performance, as the model may be biased towards the majority class.

- *Not Addressed:* This issue was not addressed using one of the traditional methods. Although I wanted to keep the focus of this thesis on the splitting technique proposed here and evaluate it, there are some techniques that can be effective for handling class unbalance in fraud detection:

  - Undersampling: Involves removing examples from the majority class to balance the class distribution. In fraud detection, this may involve randomly selecting a subset of legitimate transactions to match the number of fraudulent transactions [52].
  - Oversampling: Consists of creating additional examples of the minority class to balance the class distribution. In fraud detection, this may involve generating synthetic fraudulent transactions using techniques such as SMOTE [53].
  - Ensemble learning: Entails training multiple models on different subsets of the unbalanced dataset and combining their predictions to make a final decision. This can be particularly effective in fraud detection, as it allows the model to learn from different patterns in the data [54].
  - Cost-sensitive learning: Involves assigning a higher cost to misclassifying the minority class, which in this case would be misclassifying a fraudulent transaction as legitimate. This encourages the model to pay more attention to the minority class during training.

Poor data quality, such as missing values or outliers, can also affect model performance and limit the validity of the results.

- *Addressed:* This issue was handled during the phase of preprocessing of the dataset. It was verified whether the dataset presented missing values.

Overfitting occurs when a model is too complex and fits the training data too closely. This can result in poor generalization to new data and limit the validity of the model's predictions.

- *Addressed:* The potential issue of overfitting was addressed through the employment of a less complex neural network model, minimizing the model's propensity to capture noise and misleading patterns in the training data. This simpler model promotes better generalization to unseen data, improving robustness and reliability. The performance scores of the model were plotted over each epoch of training and validation, which allowed us to observe potential signs of overfitting such as divergence in the loss and accuracy trends between training and validation sets.

  Additionally, the *early stopping* technique was used. Early stopping is a type of monitoring of performance of the model on a validation set during training and it stops the training process once the performance starts to deteriorate, which can indicate that the model is starting to overfit.

  The modified form of k-fold cross validation technique:

  - ensures that each class has a proportional representation in each split
  - is less prone to bias due to differences in the data distribution across different splits
  - is less likely to result in overfitting since the same data is not used for both training and validation across the iterations

5.4 Prior Research

In 2022, Johari [17] proposed a study using a deep learning approach, the Convolutional Neural Network (CNN), to perform a binary classification on the PaySim

unbalanced financial dataset. Their proposal was to make an investigation where the machine learning method Restricted Boltzman Machine (RBM) applied to this same dataset, would be confronted in terms of performance, against the proposed CNN approach. The scores results showed a better performance of the CNN approach, outperforming the RBM model.

The scores of the CNN performance are shown in Table 5.7, based on the confusion matrix and compare to my proposed Full Sets approach scores:

Table 5.7: Comparison between Convolutional Neural Network vs Full Sets scores performance

Scores	**CNN**	**Full Sets**
true positives	1369	1214
false positives	17	19
true negatives	690515	552420
false negatives	702	429
accuracy	0.99890	0.99919
precision	0.98780	0.98542
recall	0.66010	0.73883
AUC	0.83000	0.99565
PRC	0.93630	0.94237
specificity	0.99990	0.99997
f1-score	0.79380	0.84367
Total transactions	692603	554082

Analysis:

- Full Sets is more proficient in identifying fraudulent transactions amongst the actual frauds than the CNN model. It also demonstrates a superior balance between correctly identifying both fraudulent and non-fraudulent transactions, and it is more effective at maintaining a balance between precision and recall.
- Full Sets and CNN has similar specificity, accuracy, precision and PRC scores, meaning that they both have the same performance in correctly identifying

non-fraudulent transactions, classifying frauds and nonfrauds, classify a transaction to be fraudulent, and have similar performances across all classification thresholds.

- Full Sets and CNN have similar false positives, false negatives and true positives.
- Although the difference in the results are not statistically significant, Full Sets has a slightly better performance than CNN.
- These unexpected results highlight aspects to be considered in regard to the dataset's properties.

5.5 Future Work & Contributions

As future work, I propose to apply regression analysis and sensitivity analysis techniques to detect confounding variables and their effect on the dependent variable on the dataset. This will enable a more comprehensive understanding of the relationship between the independent and dependent variables in the study. Through the use of regression analysis, it is possible to identify the extent to which a particular variable influences the dependent variable, while sensitivity analysis can provide insight into how changes in one variable affect the outcome of the study. These techniques will enhance the robustness of the study's findings, enabling more accurate conclusions as results. A special attention should be given to the findings related to this dataset and the possible issue it may have in regard to its representativeness as a real-world dataset.

Synthetic dataset

In 1998 and 1999, the DARPA Intrusion Detection System Evaluations [55], conducted by Lincoln Laboratory, were one of the first major efforts to evaluate the effectiveness of intrusion detection systems (IDS). These evaluations utilized a synthetic dataset for evaluation, which was generated in a military network environment and included both normal network traffic and simulated attacks.

However, the evaluations and the dataset have been subject to critique due to several reasons:

- Lack of Realism: Some critics argued that the dataset does not accurately represent real-world network traffic or attack patterns. The synthetic nature of the traffic and attacks may not capture the complexity and variability of real-world scenarios. In particular, the attacks were simulated and hence might not represent the techniques, tactics, and procedures employed by actual attackers.

- Lab Environment: The data was generated in a lab environment, which may not reflect the operational conditions in which IDS are deployed. This can affect the external validity of the evaluation i.e., the extent to which its findings can be generalized to real-world settings.

- Unbalance: The dataset has a high unbalance between normal and attack traffic, which may not be representative of the actual rates of attacks in operational environments.

- Labeling Issues: There have been concerns raised about the accuracy of the labeling in the dataset, with some instances possibly being misclassified.

- Reproducibility: Due to the unique way in which the dataset was generated, it's hard to replicate the experiments and compare the results with new IDS methods or against other datasets.

- These critiques reflect the challenges in conducting realistic and effective evaluations of IDS and highlight the importance of using representative and accurately labeled datasets in such evaluations.

A future direction for a research is to evaluate this PaySim dataset aiming the validation as a valid representative of a real-world dataset, or demonstrate the real problems with this dataset. The suggested model for this work is the convolutional neural network as the main model, CNNs have shown great potential in solving complex problems in a wide range of fields, including image and speech recognition, natural language processing, and medical diagnosis. The use of CNNs may also provide insights into the mechanisms that underlie the relationship between the independent and dependent variables.

As another future work, a direction would be to extensively test and optimize models on the PaySim synthetic dataset. The objective would be to achieve exceptional performance, significantly minimizing the rates of false positives and negatives. Once optimal results are obtained with the synthetic dataset, an intriguing next step would be to validate and compare the performance of these models on real-world datasets. Specifically, one could aim to identify and employ datasets that bear similar characteristics to the PaySim data. This could provide valuable insights into the generalizability and robustness of the models trained on synthetic data, and offer a more nuanced understanding of how effectively these models can transition from synthetic to real-world scenarios.

My contribution to the field through my thesis:

The synthetic nature of this PaySim dataset used in this thesis raises certain issues. Synthetic datasets, while advantageous in certain aspects such as privacy preservation and easy availability, have a crucial limitation: they may not completely reflect the complexities and variances present in real-world data. This discrepancy can potentially lead to models performing well in synthetic scenarios but faltering when faced with real data. Therefore, while my model's showed good results, it is important to acknowledge this limitation in interpreting its potential efficacy in a real-world context. This thesis, therefore, emphasizes the need for rigorous testing and validation of models on real-world data to truly evaluate their performance and utility in practical applications. Synthetic data aims to mimic real-world data, but depending on how it's generated, it may not perfectly capture the complexities of real-world data. This might impact the performance of the models and not be able to provide realistic results.

I recommend, in the form of a study, to evaluate the PaySim simulator with a real-world data and compare results. The techniques used to generate synthetic data must be critically examined and continuously refined to ensure they capture the key characteristics of the system they are modeling. Furthermore, models trained on synthetic data should ideally be validated on real-world data wherever possible, to affirm their effectiveness in practical applications.

In essence, synthetic datasets offer promising opportunities, but their use calls for caution, rigorous validation, and continuous refinement to ensure the efficacy of models developed using them.

The novelty in my thesis lies in a unique method of data splitting for machine learning model training. Unlike traditional approaches observed in prior works, the data is split into random parts, with an added element of shuffling, not only before the split, but also post-splitting. This additional randomness in the training dataset preparation, while seemingly simple, introduces a new level of robustness to the model's ability to learn and generalize from the data. Furthermore, the method was applied to a neural network model, representing an innovative crossover of techniques typically associated with standard machine learning algorithms.

Chapter 6

Conclusion

Financial fraud can cause significant harm to individuals, businesses, and society, including financial losses, damage to credit scores, emotional distress, and erosion of trust in financial institutions and systems. The global cost of occupational fraud is estimated to be around $3.7 trillion annually. With the rise of digital technologies, cybercriminals use various tactics to gain access to sensitive information and steal money, making it crucial for individuals and organizations to stay vigilant and take steps to protect themselves from online fraud. To combat these criminal incursions, researchers have conducted extensive studies on various aspects of fraud, including its causes, detection, prevention, and punishment.

Prior research developed relevant studies to mitigate cybercrimes, reduce statistics rates in favor of financial frauds, and keep up-to-date with the never ending advancement of criminal technology.

In the course of this thesis, I proposed a comprehensive comparison of two distinct approaches, Full Sets and Iterations, and also three different machine learning algorithms, Random forest, Logistic Regression and AdaBoost. The comparative study aimed to deliver an understanding of how these methods interact with the dataset. This examination was executed on a synthetic dataset called PaySim, which prior research has used in conjunction with convolutional neural network (CNN) approach. By undertaking such a comparative study, this research aimed to provide more extensive insights and facilitate a broader understanding of the potential of different approaches.

However, an unexpected yet significant contribution of this thesis ended up being a crucial cautionary note about the synthetic dataset used. Despite its frequent use in prior research, it became evident that this dataset might suffer from a lack

of representativeness of the real-world scenarios it aims to emulate. This lack of representativeness is a significant concern, as models trained on such a dataset might perform excellently in testing but falter when deployed in real-world applications. This revelation has potential implications for numerous studies in the field that utilize synthetic datasets and underscores the importance of rigor in dataset selection and scrutiny.

Finally, the technique used to generate the synthetic dataset raised doubts. While synthetic data can be a powerful tool when real data is scarce or sensitive, the generation method must accurately capture the key characteristics of the system it is modeling. The divergence seen between expected and observed model performance in this study suggests potential shortcomings in the data generation technique used. These concerns put forth a call to action for further investigation into the methodologies used for synthetic data generation. These insights bring value to the field, fostering critical thought about dataset choice and the potential impacts on model performance and generalizability.

Bibliography

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[2] Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc, 2019.

[3] David W. Hosmer and Stanley Lemeshow. *Applied logistic regression*. John Wiley and Sons, 2000.

[4] Classification on Imbalanced Data, 2022. URL: `https://www.tensorflow.org/tutorials/structured_data/imbalanced_data`, (Accessed on: 2023-05-09).

[5] Report to the Nations: 2018 Global Study on Occupational Fraud and Abuse, 2018. URL: `https://s3-us-west-2.amazonaws.com/acfepublic/2018-report-to-the-nations.pdf`, (Accessed on: 2023-05-08).

[6] Internet Crime Complaint Center (IC3). URL: `https://www.ic3.gov/`, (Accessed on: 2023-05-11).

[7] Internet Crime Report 2022, 2022. URL: `https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf`, (Accessed on: 2023-05-10).

[8] PwC's Global Economic Crime and Fraud Survey 2022, 2022. URL: `https://www.pwc.com/gx/en/forensics/gecsm-2022/pdf/PwC%E2%80%99s-Global-Economic-Crime-and-Fraud-Survey-2022.pdf`, (Accessed on: 2023-05-10).

[9] Cybersecurity Ventures. URL: `https://cybersecurityventures.com/`, Report for 2022: `https://cybersecurityventures.com/boardroom-cybersecurity-report/`, (Accessed on: 2023-05-11).

[10] 2022 Cybersecurity Almanac. URL: https://cybersecurityventures.com/cybersecurity-almanac-2022/, (Accessed on: 2023-05-12).

[11] The Global Risks Report 2020. URL: https://www.weforum.org/reports/the-global-risks-report-2020/, (Accessed on: 2023-05-13).

[12] Paysim: Financial Simulator. URL: https://www.kaggle.com/datasets/ealaxi/paysim1, (Accessed on: 2022-11-05).

[13] Yakub K Saheed, Moshood A Hambali, Micheal O Arowolo, and Yinusa A Olasupo. Application of GA Feature Selection on Naive Bayes, Random Forest and SVM for Credit Card Fraud Detection. In *2020 International Conference on Decision Aid Sciences and Application (DASA)*, pages 1091–1097. IEEE, 2020.

[14] Green, Brian Patrick and Choi, Jae Hwa. Assessing the Risk of Management Fraud Through Neural Network Technology. *Auditing*, 16:14–28, 1997.

[15] Hongxing He, Warwick Graco, and Xin Yao. Application of Genetic Algorithm and K-nearest Neighbour Method in Medical Fraud Detection. In *Simulated Evolution and Learning: Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98 Canberra, Australia, November 24–27, 1998 Selected Papers 2*, pages 74–81. Springer, 1999.

[16] Andreas L Prodromidis and Salvatore Stolfo. Agent-based Distributed Learning Applied to Fraud Detection. 1999.

[17] Soumya Shrivastava and Punit Kumar Johari. Convolutional Neural Network Approach for Mobile Banking Fraudulent Transaction to Detect Financial Frauds. *International Journal of Engineering Technology and Management Sciences*, 2022.

[18] Mubalaike, Aji Mubarek and Adali, Esref. Deep Learning Approach for Intelligent Financial Fraud Detection System. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 598–603. IEEE, 2018.

[19] Longbing Cao. Ai in Finance: Challenges, Techniques, and Opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[20] Michael Biehl. Supervised Learning - An Introduction, 2019.

[21] Iffat Zafar, Giounona Tzanidou, Richard Burton, Nimesh Patel, and Leonardo Araujo. *Hands-on convolutional neural networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python*. Packt Publishing Ltd, 2018.

[22] Kishan G Mehrotra, Chilukuri K Mohan, and HuaMing Huang. *Anomaly Detection Principles and Algorithms*, volume 1. Springer, 2017.

[23] PayPal Transactions Dataset. URL: `https://datarade.ai/data-categories/paypal-transaction-data`, (Accessed on: 2023-02-01).

[24] Bitcoin Transaction Dataset. URL: `https://www.kaggle.com/datasets/bigquery/bitcoin-blockchain`, (Accessed on: 2023-01-29).

[25] New York City Taxi Trip Dataset. URL: `http://https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, (Accessed on: 2023-02-05).

[26] Credit Card Fraud Detection Dataset. URL: `https://www.kaggle.com/mlg-ulb/creditcardfraud`, (Accessed on: 2023-02-10).

[27] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 3 edition, 2011.

[28] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, Inc., 2018.

[29] Raschka, Sebastian and Mirjalili, Vahid. *Python Machine Learning*. Packt Publishing Ltd., second edition, September 2017.

[30] Joel Grus. *Data Science from Scratch: First Principles with Python*. O'Reilly, 2015.

[31] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007. URL: `https://www.bibsonomy.org/bibtex/2d21de30a3a67c0f9f3c96bd6eec3267a/midtiby`, (Accessed on: 2023-01-18).

[32] Masoumeh Zareapoor, KR Seeja, and M Afshar Alam. Analysis on Credit Card Fraud Detection Techniques: Based on Certain Design Criteria. *International Journal of Computer Applications*, 52(3), 2012.

[33] G Ganesh Sundarkumar and Vadlamani Ravi. A Novel Hybrid Undersampling Method for Mining Unbalanced Datasets in Banking and Insurance. *Engineering Applications of Artificial Intelligence*, 37:368–377, 2015.

[34] Maira Anis, Mohsin Ali, and Amit Yadav. A Comparative Study of Decision Tree Algorithms for Class Imbalanced Learning in Credit Card Fraud Detection. *International journal of economics, commerce and management*, 3(12):86–102, 2015.

[35] Lopez-Rojas, Edgar and Elmir, Ahmad and Axelsson, Stefan. PaySim: A Financial Mobile Money Simulator for Fraud Detection. In *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, pages 249–255. Dime University of Genoa, 2016.

[36] José A Álvarez-Jareño, Elena Badal-Valero, José Manuel Pavía, et al. Using Machine Learning for Financial Fraud Detection in the Accounts of Companies Investigated for Money Laundering. *Economics Department, Universitat Jaume I, Castellón (Spain)*, (2017/07), 2017.

[37] Zhaohui Zhang, Xinxin Zhou, Xiaobo Zhang, Lizhi Wang, and Pengwei Wang. A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection. *Security and Communication Networks*, 2018, 2018.

[38] Sekar, M and Rengarajan, M and Manivannan, D. Credit Card Fraud Detection Using Machine Learning and Deep Learning Techniques. *International Journal of Advanced Science and Technology*, 29:2642–2648, 2020.

[39] Sumit Misra, Soumyadeep Thakur, Manosij Ghosh, and Sanjoy Kumar Saha. An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction. *Procedia Computer Science*, 167:254–262, 2020.

[40] Yara Alghofaili, Albatul Albattah, and Murad A Rassam. A Financial Fraud Detection Model Based on LSTM Deep Learning Technique. *Journal of Applied Security Research*, 15(4):498–516, 2020.

[41] Liu, Xiao and Yan, Kuan and Kara, Levent Burak and Nie, Zhenguo. CCFD-Net: A Novel Deep Learning Model for Credit Card Fraud Detection. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 9–16. IEEE, 2021.

[42] Vesta's Corp transactions dataset. URL: `https://www.kaggle.com/competitions/ieee-fraud-detection/overview`, (Accessed on: 2023-04-17).

[43] Joy Iong-Zong Chen and Kong-Long Lai. Deep Convolution Neural Network Model for Credit-Card Fraud Detection and Alert. *Journal of Artificial Intelligence*, 3(02):101–112, 2021.

[44] Daniel Kahneman, Paul Slovic, and Amos Tversky. *Judgment under uncertainty: Heuristics and biases*. Cambridge university press, 1982.

[45] Morgan Grandi. The True Cost of False Positives in The Battle to Prevent Fraud, 2020. URL: `https://www.incognia.com/blog/the-true-cost-of-false-positives-in-the-battle-to-prevent-fraud`, (Accessed on: 2023-05-27).

[46] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

[47] Online Graphreader Tool, 2023. URL: `http://www.graphreader.com/`, (Accessed on: 2023-05-28).

[48] Cook, Thomas D and Campbell, Donald Thomas and Shadish, William. *Experimental and Quasi-experimental Designs for Generalized Causal Inference*. Houghton Mifflin Boston, MA, 2002.

[49] Campbell, Donald T and Stanley, Julian C. *Experimental and Quasi-experimental Designs for Research*. Ravenio books, 2015.

[50] Dewi, Christine and Chen, Rung-Ching. Random Forest and Support Vector Machine on Features Selection for Regression Analysis. *Int. J. Innov. Comput. Inf. Control*, 15(6):2027–2037, 2019.

[51] Data, MIT Critical and Salciccioli, Justin D and Crutain, Yves and Komorowski, Matthieu and Marshall, Dominic C. Sensitivity Analysis and Model Validation. *Secondary analysis of electronic health records*, pages 263–271, 2016.

[52] Dedy Trisanto, Nofita Rismawati, Muhamad Femy Mulya, and Felix Indra Kurniadi. Effectiveness Undersampling Method and Feature Reduction in Credit Card Fraud Detection. *Int. J. Intell. Eng. Syst*, 13(2):173–181, 2020.

[53] Meng, Cuizhu and Zhou, Li and Liu, Bisong. A Case Study in Credit Fraud Detection with SMOTE and XGboost. In *Journal of Physics: Conference Series*, volume 1601, page 052016. IOP Publishing, 2020.

[54] Zhou, Zhi-Hua and Zhou, Zhi-Hua. *Ensemble Learning*. Springer, 2021.

[55] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.