MSc Thesis in Engineering & Policy Analysis

Explainable AI: A Proof of Concept Demonstration in Financial Transaction Fraud Detection using TreeSHAP & Diverse Counterfactuals

Pratheep Kumaar

August 2021

A thesis submitted to the Delft University of Technology in fulfillment of the requirements for the degree of Master of Science in Engineering & Policy Analysis

The work in this thesis was carried out in part at:

Supervisors:	Prof. Dr. ir. (B) Bert Enserink
	Dr. ir. (S) Sander van Cranenburgh
	Prof. Dr. Ir. (M.F.W.H.A) Marijn Janssen
External Advisor:	Dr. Violeta Misheva

Executive Summary

The European Commission recently published a proposal for an Artificial Intelligence (AI) act that requires the development of trustworthy AI systems for European Union markets. The proposal clearly mentions that AI systems should make use of Explainable AI tools to increase transparency and interpretability. Financial institutions in the Netherlands are required by law to detect and monitor fraud in their infrastructure. Fraud detection in Financial Services (FS) or the FinTech industry is increasingly performed by Machine Learning (ML) and Artificial Neural Network models that provide high classification performance. ML/ANN-based fraud detection systems that are necessary for maintaining trust in the Dutch financial system are classified as high-risk applications by the proposal for the EU AI act. The EU AI act will directly impact high-risk AI applications used within the EU markets, Therefore, the Dutch financial institution sponsoring this research wants to future-proof their ML-based fraud detection to improve transparency and trust by solving the model interpretability problem. Explainable Artificial Intelligence (XAI) is a domain of AI research that seeks to solve model interpretability problems of black-box ML models. In this thesis research, proofs of concepts are demonstrated for the investigation of two XAI approaches - TreeSHAP & Diverse Counterfactuals to improve the model explainability or interpretability of ML/ANN-based fraud detection systems. This research pioneers the investigation of Diverse Counterfactuals to improve model interpretability in ML/ANN-based fraud detection systems.Based on the existing literature, this is the first instance of research investigating Diverse Counterfactuals for generating explanations to ML/ANN-based fraud detection models trained using synthetic transaction datasets.

Before demonstrating the proofs-of-concept, an extensive literature survey has been conducted to map the XAI research landscape, formulate an XAI taxonomy, and conduct a comparative analysis of XAI approaches to select and describe in detail, the relevant approaches for the use-case at hand. Subsequently, several ML and ANN models have been trained and tested using the PaySim synthetic transaction datasets. To overcome model performance challenges due to data quality issues and high class-imbalance in the datasets, several experimentation scenarios involving hyperparameter optimization, SMOTE Oversampling and class-weighting have been investigated. Subsequently, two high-performing models (XGBoost & MLP) from these experiments have been used to demonstrate the proofs of concepts by investigating TreeSHAP and Diverse Counterfactual algorithms. TreeSHAP algorithm greatly improved the interpretability of the global and local model behavior of the XGBoost-based fraud detection models. Diverse Counterfactuals algorithm-generated diverse but unfeasible counterfactuals. Counterfactual algorithms suffer from computational inefficiency and therefore, further research has to be conducted to generate feasible counterfactuals. Future work on model explainability should also conduct a human-grounded evaluation of the explanations to evaluate the quality or goodness of the explanations. Finally, real-world transaction datasets should be used instead of synthetic datasets so that the research is generalizable to the real world.

Acknowledgements

The following is my thesis report on the topic "Explainable AI: A Proof of Concept Demonstration in Financial Transaction Fraud Detection using TreeSHAP & Diverse Counterfactuals". This thesis project was completed at a large Dutch financial institution in fulfillment of the requirements for the degree of Master of Science in Engineering & Policy Analysis in Delft University of Technology (TUDelft). From March 2021 until August 2021, I worked in a Data Science team within the financial institution to investigate Explainable AI approaches. I had the opportunity to conduct practical research in a dynamic, complex multi-actor work environment. In addition to researching for the benefit of both the financial institution and the university, I gained skills in Applied Machine Learning (ML) in finance, Agile workflows, Azure cloud computing, and big data processing & analytics using Apache Spark in Azure Databricks. Apart from the technical skills, I learned to work independently in a Dutch professional corporate environment with regular team meetings and presentations. I applied practical skills combined with theoretical knowledge to provide valuable insights to my team. By the end of the graduation internship, I greatly improved my time management and interpersonal skills. I leave this internship with increased confidence and knowledge to pursue professional work after graduation.

I would like to give a special thanks to my company supervisor, Dr. Violeta Misheva for guiding, supporting, and encouraging me throughout the internship project. Her support, knowledge, and inspiration greatly helped me with the successful completion of the project. I also thank my team at the financial institution especially Mrs. Mirjan Ramrattan for giving me constant support and feedback. I thank my university supervisor, Dr. Sander van Cranenburgh and the graduation committee for their timely and continuous support in the successful completion of my thesis. Finally, I would like to thank my parents and friends for their love and care. It was a humbling experience to complete this thesis during the second wave of the COVID-19 pandemic and it gave me a great sense of satisfaction throughout the process.

I hope you will get some value from reading this report.

Pratheep Kumaar
Den Haag, August 2021

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Artificial Intelligence is a comprehensive term for a large collection of intelligent technologies that are capable of mimicking human behavior like problem-solving and learning Mintz and Brodie [2019]. Several computer science techniques such as Machine Learning (ML), Artificial Neural Network (ANN), Computer Vision, and Deep Learning (DL) come under the definition of AI. According to the new EU proposal for an AI act, AI systems have the potential to contribute to the socio-economic and socio-technical well-being of a wide array of industries. AI systems make industries competitive by providing the users with predictions, the ability to optimize resources, and decision-support aia [2021]. The ability to predict outcomes and provide decision support in a complex socio-technical environment makes AI systems to be indispensable for decision-making within public and private industries. At the same time, AI systems have the potential to harm public interests, if not properly regulated. The potential for harm arises due to the black-box nature of the AI system, making the inner working of such systems hard to comprehend or interpret Adadi and Berrada [2018]. Several challenges of AI have to be addressed for successful integration of AI-driven technologies in sensitive domains such as finance and defense Gunning et al. [2019]. AI has to be seen from a socio-technical perspective as the widespread adoption of such technology is already having a significant impact on society Lentz et al. [2021].

1.1 Socio-Technical Relevance of AI

Owing to the fourth industrial revolution, the world is witness to the accelerated adoption of data-driven AI systems in almost every aspect of our society. Global investments in AI are expected to reach more than 50 billion U.S. dollars by 2021, according to International Data Corporation Adadi and Berrada [2018]. AI systems make important decisions in our daily lives from product recommendations in e-commerce sites to making high-risk financial decisions in banks. The critical decisions made with the help of AI systems affect human lives in healthcare, finance, law, and autonomous driving Arrieta et al. [2019]. The decisions and decision support provided by these AI systems not only impact our individual choices and behaviors but also have a significant impact on large-scale socio-technical systems. Moreover, these AI systems are inherently sociotechnical systems as they are formed by subsystems of models, data, and human interactions Bhatt et al. [2020] Lawrence [2019]. One of the important socio-technical systems where AI has seen widespread growth is finance Cao [2020].

1.2 AI in Finance

Over the last several years, interest in the adoption of AI in the financial sector has seen exponential growth. Figure 1.1 shows the exponential growth in Google search trends for terms such as "AI in Finance" and "Data Science in Economics" Cao [2020]. The infusion of modern digital technologies such as AI in the financial sector has given rise to the new field of "FinTech" Zhang and Kedmey [2018]. The contributions of AI in FinTech are widespread and spans various domains of the financial sector. The contributions of AI are seen at a high-level socio-economic paradigm involving financial simulations and modeling, financial optimization, and financial ethic assurance. On the other hand, the contributions of AI are seen at a low-level business-specific paradigm involving financial product recommendation, market forecasting, investment allocation, and financial fraud detection Cao [2020]. Figure 1.2 shows the various high-level socio-economic and low-level business-specific contributions of AI in FinTech Cao et al. [2020]. In the Netherlands, the Dutch financial regular - De Netherlandse Bank (DNB) four crucial areas for AI adoption - customer-focused applications, operations, investment

management, and regulatory compliance Van der Burgt [2019] FSB [2017]. Van der Burgt [2019] recognizes that the Dutch financial institutions are starting to adopt AI for ensuring regulatory compliance and safeguarding customer's interests by the use of AI systems in fraud detection and transaction monitoring. The contribution of AI technologies such as Machine Learning (ML) in financial fraud detection and transaction monitoring has been significant.

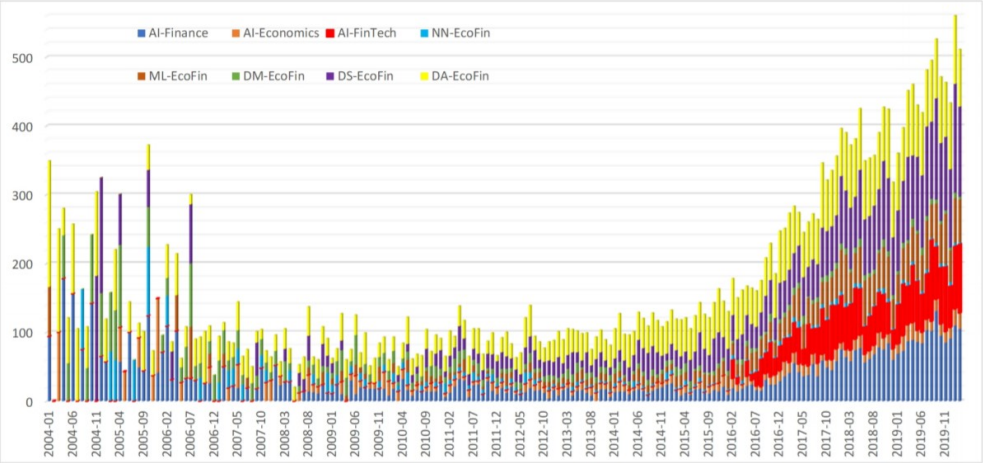


Figure 1.1: An Overview of Google search trends for AI and Data Science/Data Analytics (DS/DA) in Finance and Economics. The search engine hits for AI in Finance has exploded in the last several years. Cao [2020]

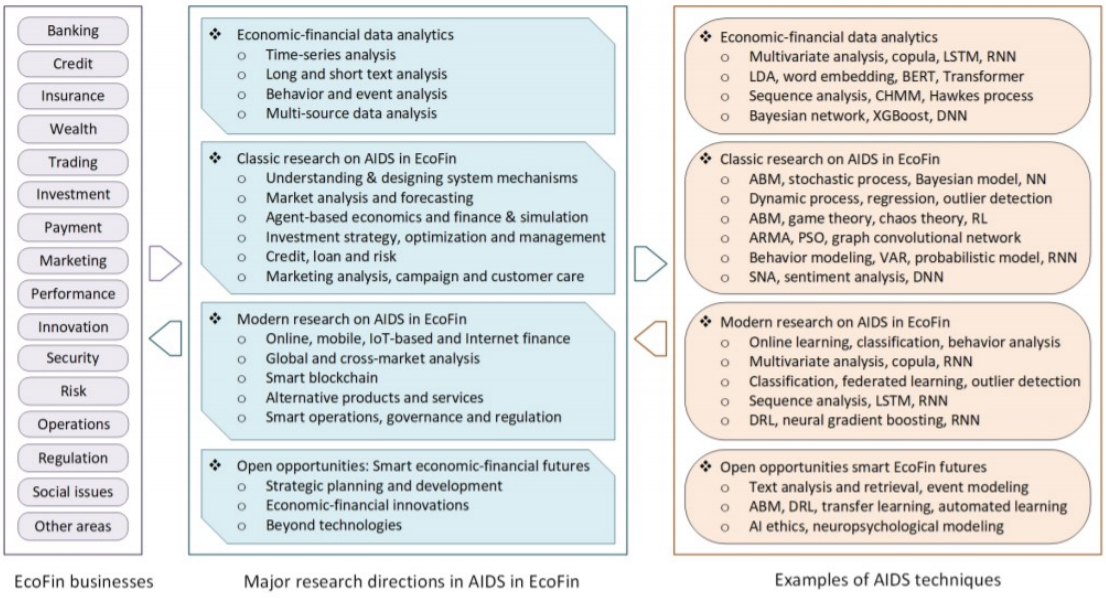


Figure 1.2: An overview of various AI/Data Science (AIDS) research directions towards enabling smart decision-making in the various domains of Eco-Fin (Economic/Finance) businesses. Cao et al. [2020]

1.3 Fraud Detection in FinTech

Due to the increasing adoption of digital technologies by businesses and entrepreneurs, online payment systems are widely used for conducting financial transactions. The online financial transactions

facilitated by the digital payment systems are faster than traditional transaction systems Syahadiyanti and Subriadi [2018]. To no surprise, online transaction fraud has increased along with the increasing adoption of digital payment systems. According to UK Finance, the total losses in 2019 due to financial fraud in the UK amounts to £828.8 million, and in that losses due to online transaction fraud through mobile payment systems amounts to £455.8 million Worobec [2020]. During the COVID-19 pandemic, there has been a 70% increase in online fraud [1] with £34.5 million stolen from innocent customers in the UK alone [2]. Moreover, financial institutions faced regulatory fines to the tune of $321 billion over the last decade globally for non-compliance with fraud detection, anti-terrorism funding, and anti-money laundering regulations Gade et al. [2019]. Fraud detection has become crucial for financial institutions globally due to the need for protecting customer's interests and preventing regulatory sanctions. To detect and prevent online transaction fraud, financial institutions utilize fraud detection systems in their transaction monitoring process. Fraud detection systems are autonomous systems that detect anomalous or suspicious transaction flows between two or more entities within the banking infrastructure. The fraud detection systems have traditionally been rule-based Kou et al. [2004] but increasingly Machine Learning (ML) models are used nowadays Zhang and Trubey [2019] due to its various advantages.

Rule-based fraud detection systems are autonomous "Expert Systems" that use knowledge from domain experts to detect fraudulent transactions Kou et al. [2004]. The expert knowledge is structured into a series of if-else conditionals and transactions are classified into genuine or fraudulent, based on this set of rules. Figure 1.3 shows a system-level model of a digital rule-based fraud detection system. The inference engine of the fraud detection system uses a knowledge base of if-else conditionals to generate fraud alerts Ahmed et al. [2021]. The main advantage of rule-based systems is that they are simple and easy to interpret but they bring along several challenges for the fraud detection tasks. One of the main challenges is that rule-based systems are static and don't recognize emerging patterns of fraud or money laundering. Rule-based systems also provide lack-lustre performance in detecting fraud leading to high False Negative Rate (FNR) Kou et al. [2004] & Modi and Dayma [2018]. FNR is the percentage of fraudulent transactions that are incorrectly identified as genuine. An increase in FNR of a fraud detection system within the financial institutions can lead to regulatory sanctions Gade et al. [2019].

1.4 Related Work

Due to the challenges with rule-based systems, the research into the application of ML in fraud detection has seen considerable interest in the academia Deng et al. [2021]. According to Phua et al. [2010], both supervised and unsupervised ML methods can be effective in detecting fraudulent transactions. Zhang and Trubey [2019] shows empirical evidence that ML methods such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Artificial Neural Network (ANN) provide improved performance in detection fraud and money laundering. Among the methods researched by Zhang and Trubey [2019], ANN models provide the best performance.

Several novel ML approaches have been proposed for improving fraud detection in the FinTech industry in recent years. Nami and Shajari [2018] proposes pioneers the use of dynamic random forest algorithms for cost-sensitive fraud detection. Makki et al. [2017] conducts a comparative analysis of several supervised and unsupervised ML techniques for fraud detection tasks. Abdulla et al. [2015] proposes a hybrid two-stage approach for fraud detection using a Genetic Algorithm (GA) for feature engineering and subsequently Support Vector Machines (SVM) for generating fraud alerts. Zareapoor and Yang [2017] introduces a "balancing strategy" for mitigating the performance challenges faced by ML fraud detection models trained using class-imbalanced transaction datasets.

[1] https://www.telegraph.co.uk/news/2021/05/13/online-fraud-70-per-cent-covid-pandemic/
[2] https://www.bbc.com/news/technology-56499886

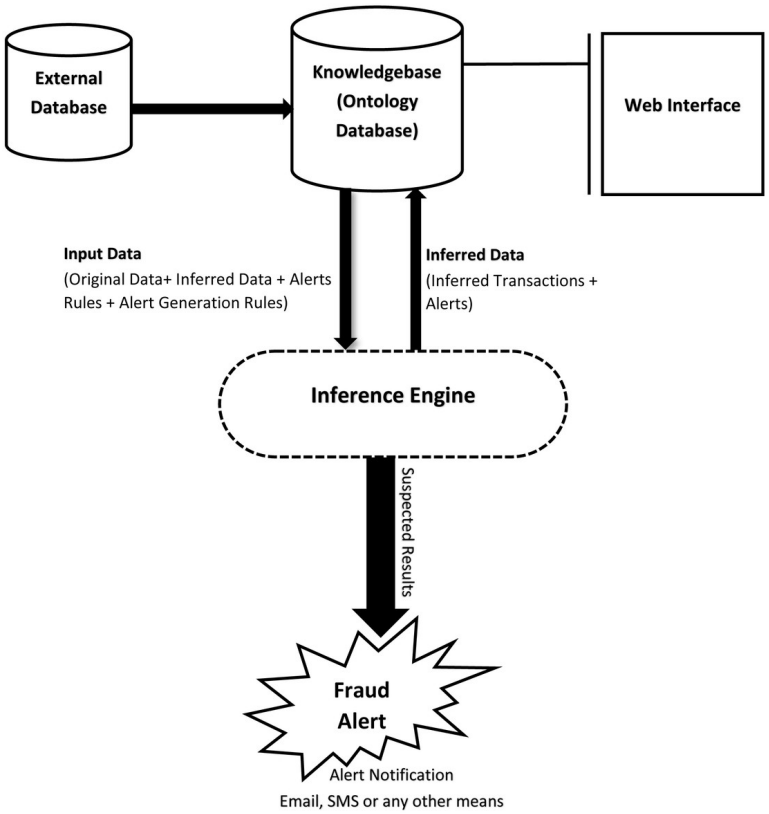


Figure 1.3: An overview of system model of a rule-based fraud detection system where the inference engine uses a knowledge base of if-else conditionals to generate fraud alerts Ahmed et al. [2021].

1.5 Interpretability Problem

Although the integration of ML in fraud detection systems has improved their performance, there are serious concerns regarding the model explainability or interpretability Gunning et al. [2019]. The concerns arise due to the black-box nature of the ML and ANN models, wherein the inner working of the models are not fully known to the model developers or users Arrieta et al. [2019] & Guidotti et al. [2018]. According to the EU proposal for an AI act, lack of interpretability in the ML models used in financial institutions can have the potential to harm public interests if not properly regulated. These concerns have led governments and financial regulators across the world to formulate AI policies and regulatory guidelines for the responsible use of AI in Financial Services (FS). In the Netherlands, the government recognizes this interpretability problem and in 2019, devised a national AI policy called "Strategic Action Plan for AI". The Dutch AI policy seeks to mitigate the interpretability problem by encouraging the research into the development and adoption of responsible AI practices Dut [2019]. Also in 2019, the Dutch financial regulator - DNB came up with a framework of guiding principles for the use of AI in the financial sector Van der Burgt [2019]. It is clear from the regulator that any use of AI in the Dutch financial sector should focus on the "explainability, simplicity, and reliability" of AI systems in addition to system performance. Moreover, the proposal for an upcoming European Union AI act, makes clear that the adoption of AI in the financial sector should involve the use of "interpretation" methods and techniques aia [2021]. The ML interpretation methods and techniques in the scientific literature are collectively termed as Explainable AI (XAI) Adadi and Berrada [2018]. According to Arrieta et al. [2019], XAI is a field of AI research that seeks to develop techniques that can,

- Explain its model behavior and outcome while not compromising on system performance (Model Accuracy).
- Help model users interpret, comprehend and trust the model behavior and outcome of AI systems.

1.6 Explainable AI in the Sponsoring Financial Institution

The institution sponsoring this research is one of the largest financial institutions in the Netherlands and it provides financial services to all sections of Dutch society. It has started adopting AI in high-risk business processes such as financial fraud detection, investment management, and automated loan processing for its clients. Given the continuously evolving legal and regulatory environment around AI governance in the Netherlands, the financial institution wants to enhance its Explainable AI capabilities as part of its broader AI strategy. AI model developers and model users within the institution see a greater need for XAI in their AI use-cases. Technical stakeholders such as AI Engineers and Data Scientists believe XAI can help with AI model debugging. On the other hand, business stakeholders who use AI for their decision-making, believe XAI can help with explaining AI-driven decisions to their clients. Business stakeholders also believe that XAI can enable them to be better decision-makers. Moreover, the financial institution believes in future-proofing its AI systems. Therefore, the institution wants to enhance its XAI capabilities for critical AI use-cases like online transaction fraud detection. Due to the stringent regulatory measures around financial fraud detection in the Netherlands, the institution wants to adopt sophisticated ML models for financial fraud detection and anti-money laundering (AML). It is therefore essential for the institution to solve the interpretability problem with fraud detection models to enable trust with their customers, comply with current and upcoming AI regulations, and lead AI innovation in FinTech.

1.7 Research Objective

To overcome the interpretability problem and enable the adoption of advanced ML techniques in fraud detection systems in the financial institution sponsoring this research, this thesis aims to investigate XAI approaches for ML & ANN-based fraud detection systems. As proposed by the scientific literature Arrieta et al. [2019], Lundberg et al. [2019], & Gunning et al. [2019], XAI approaches can improve the explainability or interpretability of ML-based systems to overcome the interpretability problem. This thesis research aims to investigate the domain of fraud detection by accomplishing the following research objective,

"To investigate local and global, model-agnostic post-hoc explainability as a proof-of-concept for improving explainability or interpretability in Machine Learning and Artificial Neural Network (ANN) models used for online transaction fraud detection"

In Chapter 3, a detailed description of the research problem and the research framework intended to accomplish the main research objective is provided.

1.8 Research Contribution

This thesis research has both academic relevance to the broader XAI scientific community and practical relevance to the financial institution sponsoring this research. One of the main scientific contributions of this research is the proofs of concepts demonstration of XAI in improving model explainability or interpretability of fraud detection systems used in the FinTech industry, This research also seeks to provide practical guidance to the sponsoring financial institution in terms of realizing a real-world implementation of the XAI approaches investigated in this research. This research pioneers the investigation of Diverse Counterfactuals to improve model interpretability in ML/ANN-based fraud detection systems. Based on the existing literature, this is the first instance of research investigating Diverse Counterfactuals for generating explanations to ML/ANN-based fraud detection models trained using synthetic transaction datasets. Before demonstrating the proofs-of-concept, an extensive literature survey has been conducted to map the XAI research landscape, formulate an XAI taxonomy, and conduct a comparative analysis of XAI approaches to select and describe in detail, the relevant approaches for the use-case at hand. Subsequently, several ML and ANN models have been trained and tested using the PaySim synthetic datasets. To overcome model performance challenges due to data quality issues and

high class-imbalance in the datasets, several experimentation scenarios involving hyperparameter optimization, SMOTE Oversampling and class-weighting have been investigated. Upon demonstrating the proofs-of-concept using feature contribution method - TreeSHAP and example-based method - Diverse Counterfactuals, the feasibility of improving model agnostic post-hoc explainability of ML/ANN-based fraud detection models have been established. Furthermore, a practical implementation of the XAI approaches has provided sufficient insights for generalizing the findings to real-world use-cases within the financial institution sponsoring this research albeit the nature of such real-world implementations will be more complex. In addition to the financial institution, the practical insights from this thesis research could be interesting to the Dutch financial regulator - DNB in terms of knowledge sharing and regulatory oversight with the wider FinTech audience in the Netherlands.

1.9 Thesis Outline

The outline of the thesis report is as follows,

- Chapter 2 describes the background information on Machine Learning (ML) and Explainable AI (XAI) concepts and terminologies.
- Chapter 3 describes the context of the research problem. The research framework is also formulated along with the discussion of research methods.
- In Chapter 4 several Explainable AI approaches relevant for fraud detection systems are selected from the literature and described along with their implementation pseudo-code.
- In Chapter 5 the synthetic data required for developing and evaluating fraud detection models are analyzed and engineered. Two new features are artificially engineered to account for data quality issues.
- In Chapter 6, several Machine Learning (ML) and Artificial Neural Network (ANN) based fraud detection models are developed under different experimentation scenarios.
- In Chapter 7, the fraud detection models are evaluated for their performance and the model results are summarised.
- In Chapter 8, proofs of concepts of XAI are demonstrated to improve the interpretability of ML/ANN-fraud detection models using TreeSHAP and Diverse Counterfactuals.
- In Chapter 9, the conclusion along with limitations and future work are discussed. In addition to the conclusion, academic, socio-technical, and recommendations to the financial institutions are discussed.

2 Background to Machine Learning and Explainable AI

In this chapter, various terminologies, concepts, and definitions in the field of Machine Learning (ML) and Explainable AI (XAI) are introduced and elaborated. Section 2.1 gives an introduction to Machine Learning (ML) and Artificial Neural Network (ANN) models used for detecting fraud in transaction data in Chapter 6. In section 2.2, an elaborate introduction to XAI terminologies and concepts are provided to put forward a theoretical perspective of the XAI, which helps in the practical demonstration of a proof-of-concept of XAI in Chapter 8.

2.1 Machine Learning

According to the new EU proposal for an Artificial Intelligence (AI), act aia [2021], AI is an autonomous system with varying degrees of autonomy that is capable of solving human-defined objectives by providing predictions, recommendations, and another decision-support tooling. Machine learning (ML) is a class of AI systems that seeks to provide outputs to human-defined objectives by fitting a model predictive function to the available information. According to Bishop [2006], ML is simply a computer program that can learn and improve on a specific task (T) based on learning experience (E) from data sources. ML models continuously improve their performance on T by capturing E in a dynamically changing environment. The process of the learning experience (E) from data is called model training. The data used for learning E is called training or sample data. ML models do not require explicit programming to perform any decision task. It requires minimal intervention to support data-driven decision-making, unlike traditional computer programs that depend on explicit programming of decision logic. ML models generate predictions to enable decision-making based on learning experience from historical training data. The process of generating predictions from newly available data is called model inference. Mathematically, the trained ML models are essentially complex model prediction functions that transform a specific set of inputs to desired outputs Bishop [2006]. The following section gives an overview of the ML types and techniques.

2.1.1 Machine Learning Types

The following are the different types of ML,

- **Supervised learning**: The supervised learning type uses labeled training data to infer a model prediction function that can transform inputs to desired outputs. When the prediction target outcome is categorical, then it is a classification task, and if it is continuous, then it is a regression task Suresh et al. [2013]. In supervised learning, the error (difference between the initial predicted output and known labeled output) is minimized in the model training process. In the model training process, the model can be trained to classify between two classes of output (binary classification task) or more than two classes of output (multi-class classification task). Logistic Regression, Decision Tree, Random Forests, XGBoost are the most commonly used supervised learning methods. Some examples of supervised learning tasks are email spam filtering, image recognition, medical diagnosis, fraud detection, and recommendation systems. Gianey and Choudhary [2018].
- **Unsupervised learning**; The unsupervised learning type infers patterns in unlabeled training data Alloghani et al. [2020]. Clustering, Anomaly Detection, and Principal Component Analysis (PCA) are the most commonly used unsupervised learning methods El Naqa and Murphy [2015].

- **Reinforcement learning**: In reinforcement learning, environment-aware agents undertake tasks to maximize profits or total reward El Naqa and Murphy [2015].

2.1.2 Machine Learning Techniques

There are several techniques associated with the above-mentioned ML types, and they are classified into two major groups - ML and Artificial Neural Networks (ANN).

- **Artificial Neural Network (ANN)**: ANN is modeled after a biological neural system. In a simplistic form, ANN has a layered network structure of input, hidden, and output layers. These simple network structures called Multi-Layer Perceptron (MLP) are used for supervised classification or regression tasks. The layers consist of neurons or nodes that transmit signals between the input and output layers Mubalaike and Adali [2017]. A detailed introduction to the working of ANN is provided in Section 2.1.3. There are various complex architectures of ANN used for several different tasks ranging from image classification to advanced speech analytics.
- **Shallow ML**: Any ML technique that does not have a layered network structure is regarded as the Shallow ML technique. The term "shallow" denotes the lack of interconnected layered network architecture. The most commonly used Shallow ML techniques are Decision Trees (DT), Ensemble models, and Support Vector Machines (SVM) Suresh et al. [2013]. The ensemble models are further grouped into bagging and boosting techniques. Random Forest and XGBoost are popular bagging and boosting methods respectively. In the literature, shallow ML techniques are generally referred to as ML techniques. To avoid confusion and to align with the scientific literature, shallow ML techniques have been referred to as ML techniques throughout this thesis. In the literature, the term "shallow" is used to distinguish non-neural network-based ML techniques from ANN. Therefore, for simplicity, shallow ML techniques are referred to as ML techniques in this thesis research.

2.1.3 Algorithmic Definition of ML Techniques

A detailed introduction to the algorithmic definitions of ML and ANN techniques used in this thesis research is provided below. Every ML and ANN technique discussed in this section has been used to develop fraud detection models in Chapter 6. A basic algorithmic understanding of the techniques is essential to identify and discuss the advantages and disadvantages of using these techniques for fraud detection.

Multi-Layer Perceptron (MLP)

A Multi-layer Perceptron (MLP) is a type of feed-forward ANN Mubalaike and Adali [2017] that has three layers of nodes or neurons. MLP is called a "vanilla" neural network when they have only one hidden layer. As shown in Figure 2.1, the architecture of the MLP is composed of input, output and hidden layers Mubalaike and Adali [2017]. MLP is a feed-forward network so the node-level activation happens from input to output layers. The node activation is governed by an activation function such as signum or reLU. MLP is a non-linear estimator since any weighted interconnected network is not suitable for linear estimation. MLP is suitable for supervised learning tasks where the model training is done through backpropagation. Backpropagation is the process by which the weights associated with each interconnection between the nodes are updated by minimizing a multi-dimensional loss function.

let us consider weight w_{jk} between node j and k. For m inputs, the activation at k in the hidden layer is given by the below equation Mehlig [2019],

$$a_k = \sum_{j=1}^{m} w_{j,k} a_j + b_k$$

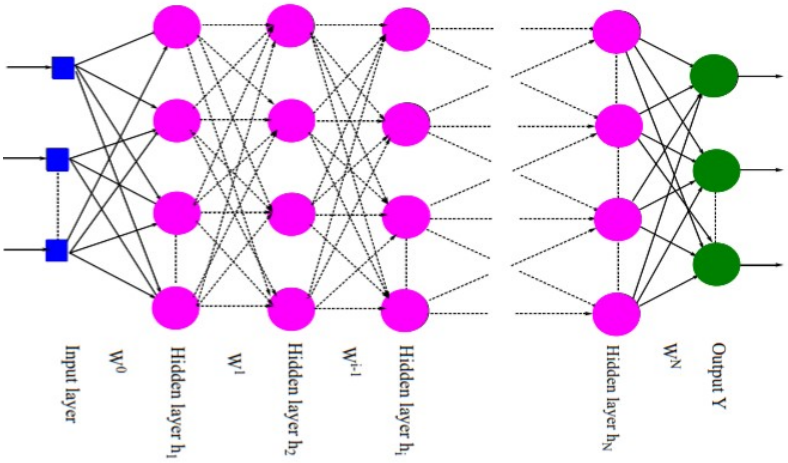


Figure 2.1: **A Graphical Overview of a MLP Architecture showing the Input, Hidden and Output Layers** - The input layer and output layers are connected to each other through a network of interconnected hidden layers. The connections between the nodes in two different layers are weighted to represent the strength of the connection Ramchoun et al. [2016]

Logistic Regression

Logistic regression or logit model is a classification model that models "the probability of a certain class" within a probability range of $[0, 1]$. In a binary logistic regression model, two dependent variables are representing two classes either 0 or 1. In the binary logit model, the logarithm of the odds (log-odds) for a dependant variable is a linear combination of one or more independent variables.

let us assume a linear relationship between independent variables a_1, a_2 and the log-odds of one class. In here, the logistic function is,

$$l = log_b \frac{p}{1-p} = \beta_0 + \beta_1 a_1 + \beta_2 a_2$$

where $p = \frac{e^l}{1-e^l}$, β is the model parameter and b is the logarithmic base.

Linear SVM

Support Vector Machines (SVM) are essentially binary linear classifiers. SVM classifies data points into distinct classes using support vectors by maximizing the distance (optimal margin) between the support vectors Chauhan et al. [2019]. Support vectors signify the borderline instances of the linearly separable input data that distinguish one class from the other as shown in Figure 2.2. For non-linear input data, mathematical functions called kernels are used to transform input data to a high dimensional feature space where hyperplanes can separate the data linearly. Hyperplanes are high-dimensional support vectors that distinguish one class from other in higher dimensions as shown in Figure 2.3.

Let x_1 and x_2 be two input data points, K be the "kernel function" Chauhan et al. [2019] that transform lower dimension input to higher dimension feature. "ϕ is the mapping applied on the input space" Chauhan et al. [2019]. The kernel k is given by the below equation,

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

If input space is equal to the feature space, then the "mapping ϕ is identity mapping (i.e) $\phi(x) = x$" Chauhan et al. [2019]. In this case, the kernel K is a linear kernel, and SVM is called a linear SVM.

Linear SVM provides high classification performance in applications using high-dimensional feature space Chauhan et al. [2019].

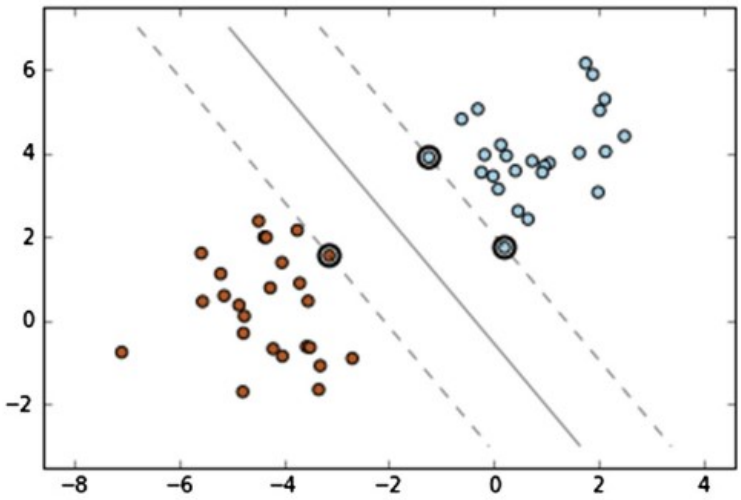


Figure 2.2: SVM finds the optimal margin between the support vectors. The dotted line passing through the circled data points are the support vectors.Chauhan et al. [2019]

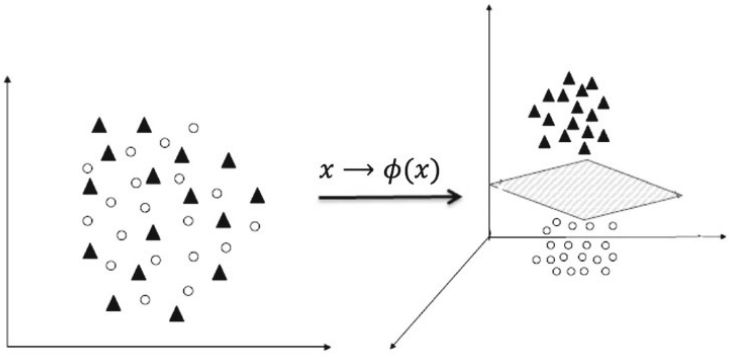


Figure 2.3: The kernel function transforms the input space (left) to high-dimensional feature space (right). Left part has two-dimensional inseparable non-linear data while the transformed feature space in higher dimension can be separated by hyperplanes Chauhan et al. [2019].

Ensemble Learning

Ensemble Learning is a class of ensemble methods that group multiple classifiers or regressors to predict an outcome. The classifier or regressor can be trained using any supervised ML models such as *Decision Tree, Neural Network or Linear Regression*. Ensemble learning for a classification task aims to combine classifiers so that the classification error of an ensemble of classifiers is much lesser than the error of a single classifier Sagi and Rokach [2018]. Ensemble methods have several advantages over a standalone classifier. For example, ensemble methods are good at avoiding overfitting, ensemble methods are computationally efficient than standalone classifiers and ensemble methods can handle data with poor class representation. Ensemble methods can mitigate the following challenges in ML-based fraud detection systems,

- ***Class Imbalance:*** Class imbalance happens when the different classes within the dataset are not equally represented. For example, if the fraudulent transactions in a transaction dataset are far fewer than the genuine transactions, then the dataset represents a class imbalance. According to

Sagi and Rokach [2018] & Galar et al. [2012], tree-based ensemble methods have a high classification performance with class-imbalanced datasets.

- ***Concept Drift:*** In real-world ML applications, features and labels drift away from their original distribution over time. Ensemble methods can be used to mitigate concept drift.

- ***Curse of dimensionality:*** ML models find it difficult to converge to a solution and avoid overfitting when handling high-dimensional feature space. It is called the "curse of dimensionality". Ensemble methods such as bagging can reduce the problems caused by the 'curse of dimensionality.

There are two main methods in ensemble learning - *Bagging and Boosting*, that have shown to provide better classification performance in handling class-imbalanced datasets Galar et al. [2012].

Bagging or bootstrap aggregation is an ensemble method known to improve 'accuracy' and 'stability of classification tasks Sagi and Rokach [2018]. Bagging is good at avoiding overfitting reducing variance in model output. Random Forest is a popular bagging technique, which uses a large ensemble of 'independent, unpruned, random decision trees' to increase the classification accuracy. Decision Trees are a tree-like classifier where split rules are learned at the nodes or leaves from the training data. Random forests have a simple theoretical foundation but can provide good classification performance. This bootstrap method trains random decision trees on random subsamples of the training data with replacement where-in the decision tree splits are randomized. The best splits at the nodes in the decision trees are not based on the 'entropy gain measure' but probabilistically proportional to the value of the variable at the nodes. The implementation of the Random Forest is shown below Sagi and Rokach [2018].

Algorithm 2.1: Implementation algorithm of Random Forest

Data: Training data S
Input : A Decision Tree DT, Number of Iterations T, subsample size μ, number of variable at DT node N
Output: $M_t : \forall t = 1, ..., T$

```
for t ∈ {1, ..., T} do
    S_t = Create subsamples μ from S with replacement
    Train a classifier M_t using DT(N) on S_t
    increment t (t++)
end
```

Boosting is another ensemble method that can reduce both bias and variance in the model output. Boosting algorithms iteratively learn weak classifiers (classifiers with poor accuracy) and add them to a strong classifier. Each iterative addition of a weak classifier to strong classifier results in the readjustment of the model-weights and thus the term 'Boosting'. *XGBoost* is a '*Gradient Tree Boosting*' algorithm and it is very popular as a distributed, scalable high-performance ML algorithm Chen and Guestrin [2016]. LightGBM is another popular '*Gradient Tree Boosting*' algorithm. It is called "Light" because it uses less memory to make predictions. Bagging techniques like Random Forest make use of smaller ensembles of decision trees but grows the trees to the maximum depth in the learning process. On the other hand, Boosting techniques grow larger ensembles of decision trees but have fewer branches with only a few splitting conditions at the nodes (as shown in Figure 2.4). Due to this, Boosting techniques suffer from overfitting. Due to the smaller ensemble of trees with shorter branches, Boosting techniques like XGBoost are interpretable while delivering high performance. XGBoost mitigates overfitting by regularisation. Regularisation can be considered as applying a penalty to force the optimization function to generalize the results well or find the unique optimal solution. The code-level implementation of the XGBoost algorithm is out of scope for this thesis. A generalized implementation of gradient boosting algorithm and XGBoost is given in the paper Chen and Guestrin [2016].

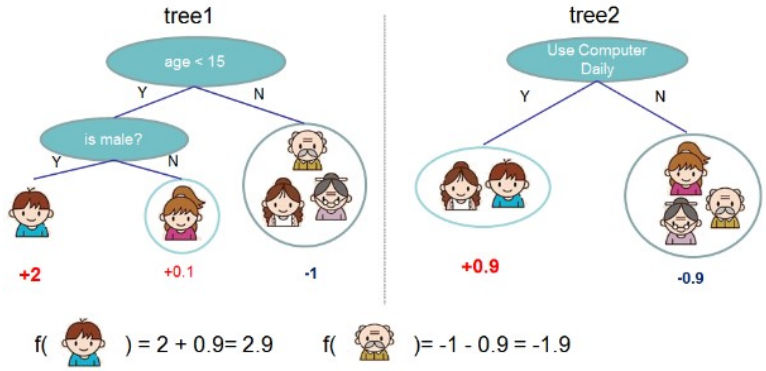


Figure 2.4: Tree-based ensembles like XGBoost calculate predictions for each instance by aggregating the predictions from each decision tree within the ensemble of trees for the specific instance Chen and Guestrin [2016]. In this example, the XGBoost prediction is the aggregate of the branch values in tree1 and tree2. By aggregating the predictions, the final prediction score that a male under the age of 15 to use computers daily is 2.9 while the final prediction score that someone above 15 who uses a computer daily is -1.9

2.2 Definition of Explainable AI

There is no clear agreement in the scientific community on a single definition for XAI. XAI is a rapidly developing field with continuously evolving terminologies, definitions, and concepts. The definition of XAI can change depending on the system modalities, use case, and the stakeholders involved. Any definition of XAI for the domain of fraud detection in Financial Services (FS) should include the multi-actor environment under which the system operates. Given this reasoning, the definition from Arrieta et al. [2019] suits the overall theme of this thesis research and it is as follows,

"Given an audience, an Explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand."

Therefore, explanations to AI-based decision-making are dependent on the stakeholder who is receiving the explanation. The notion of explanation is not absolute but relative to the context and the stakeholders interacting with the AI systems. The type of explanation will change for every stakeholder role. For example, a fraud detection analyst would need a type of explanation different from a model developer who develops the fraud detection model. According to Gunning et al. [2019], the field of Explainable AI (XAI) tries to produce explainable and interpretable models that help users to understand and effectively trust the new generation of black-box AI systems. The target of an XAI system is an explainee who makes decisions based on the decision support provided by the AI system. For example, this could be a fraud detection analyst in a financial institution who determines if a specific transaction is fraudulent or not, based on the decision support provided by an AI system.

2.2.1 Explainable AI Conceptual Model

There are many differences in the definition of the key terms related to XAI - model explainability & interpretability. They are often used interchangeably in the scientific literature Hall et al. [2019] Gunning et al. [2019] Adadi and Berrada [2018]. According to Arrieta et al. [2019], interpretability is the ability to explain AI-based decision-making in a meaningful, understandable, and comprehensible way. On the other hand, explainability is the explanatory interface between a human and an AI system where the explanation medium is human-comprehensible. Interpretability is a "passive characteristic" of a model while explainability is an "active characteristic" of a model Arrieta et al. [2019]. To avoid confusion and to align with the vast majority of XAI literature, explainability and interpretability have been

used interchangeably in it's meaning within this research. They convey the same meaning of providing transparency or clarity to AI systems discussed in this research.

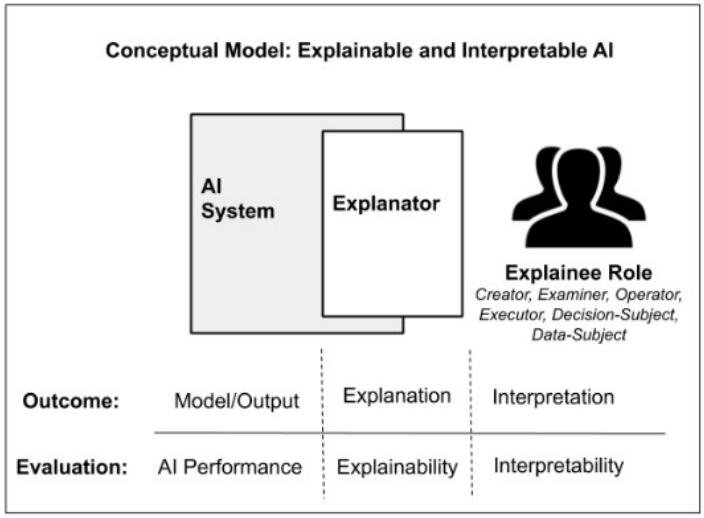


Figure 2.5: XAI Conceptual Model showing three important components in any XAI system - AI system, Explanator & Explainee

Figure 2.5 shows an XAI conceptual model that compartmentalizes an XAI system into three components- the AI system, the explanator, and the explainee Hall et al. [2019]. The explanator generates an explanation depending on the requirements of the explainee who receives the explanation to the AI system behavior or output. The outcomes of an XAI system are both AI model output and the explanation output from the explanator. Both the performance of the AI model and the explanator have to be evaluated to assess the performance of an XAI system Hoffman et al. [2018]. Here, interpretability refers to the cognitive capacity of the explainee to interpret the explanation provided by the explanator. On the other hand, explainability refers to the ability of the explanator to explain AI system behavior to the explanator.

2.2.2 What is an Explanation?

Explanations are an important aspect of any XAI system. To a stakeholder developing or using an AI system, explanations are a bridge between the opaqueness and transparency of an AI system. In general, an explanation is an answer to questions of explainability, accountability, transparency, interpretability, and auditability of AI systems Arrieta et al. [2019]. According to Mittelstadt et al. [2019], an explanation is an answer to the following questions,

- "Is the system working as intended?"
- "Do the decisions being made seem sensible?
- "Are we conforming to equality of regulation and legislation"
- "Am I being treated fairly?"
- "What could I do differently to get a favorable outcome next time?"

"Explanation Sciences" is a well-established field in philosophy, social sciences, and law while explanation in the context of AI is a relatively novel domain of research. Therefore, it is imperative to define the key features of an explanation in the context of AI systems and not solely borrow the definition of explanation from social sciences or philosophy.

The following are the key features of explanations in the context of AI,

- An explanation depends on the AI use-case at hand and therefore, it is very contextual to the AI use-case and whom it is intended for.
- Explanations are not absolute but they are relative Arrieta et al. [2019]. It is dynamic and needs to be tailored to the requirements of different stakeholders (For example, in FS - Data Scientists, Model validators, Legal & Compliance, and Business Managers).
- Explanations can be communicated through different media formats such as visual, textual, and audio Anjomshoae et al. [2019].
- Explanations are not always necessary and can be counterproductive (For example, explanations about the working of a financial crime detection model to certain stakeholders can lead to security risks)
- Explanations cannot be provided for every use-case (For complex ANN, explanations are still hard to achieve.).
- Explanations can be generated in a multitude of diverse ways using many different approaches. Murdoch et al. [2019].
- Providing explanations requires the design of an "explanation process" Hall et al. [2019] & Langer et al. [2021].
- Explanations can be evaluated on their goodness or quality Hoffman et al. [2018].

2.2.3 Taxonomy of Explainable AI

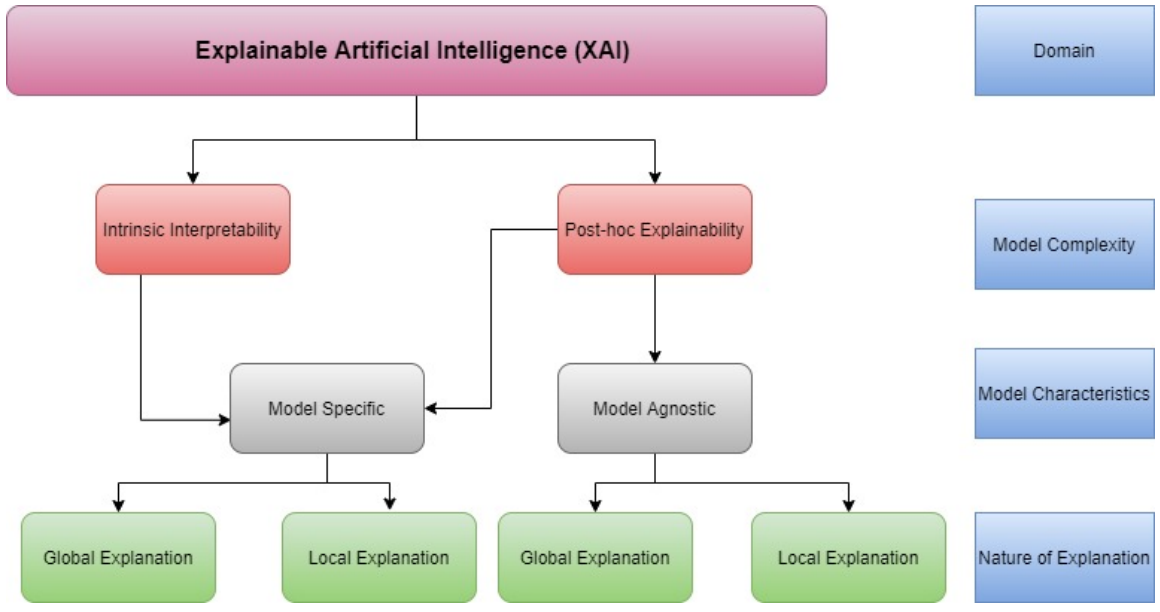


Figure 2.6: Taxonomy of XAI approaches - XAI approaches are mainly grouped into two categories such as Intrinsically interpretable and Post-hoc explainable approaches. Arrieta et al. [2019] & Adadi and Berrada [2018]

Explainability or interpretability in AI systems such as ML models is addressed by several algorithmic techniques Arrieta et al. [2019]. These techniques, methods, and tools can be collectively called the XAI approach Langer et al. [2021]. A taxonomy of XAI approaches has been illustrated in Figure 2.6 to help understand the current landscape of XAI research. The XAI approaches are classified according to the following three criteria,

- **Model Complexity**: Based on the complexity of the ML models, they are classified as white-box or black-box models. The former is generally interpretable but lacks model performance and the latter is complex and opaque as their inner model mechanisms are unknown to the user. Black box models have high model performance but lack explainability or interpretability Adadi and Berrada [2018]. Depending on the complexity of the model, explainability is achieved through inherently interpretable or post-hoc explainability approaches. White box models are made inherently interpretable during the training process of the ML model. Post-hoc explainability denotes adopting explainability after the training process of ML models. White box models are associated with inherently interpretable approaches while black-box models are associated with post-hoc explainability approaches. A Decision Tree (DT) is inherently interpretable while an ANN-based image classifier can be explainable or interpretable with post-hoc explainability approaches like feature contribution or example-based methods Arrieta et al. [2019].
- **Model Characteristics**: Depending on the characteristics of the ML model, XAI approaches can be model agnostic or model specific. Model-specific explainability approaches are specific to a certain type of ML model while model agnostic approaches apply to different types of models Molnar [2020]. According to Adadi and Berrada [2018], white box models that are inherently interpretable, generally use a model-specific XAI approaches. Post-hoc explainable models are generally model-agnostic.
- **Nature of Explanation**: Depending on the scope of the explanations needed, the nature of the explanation can be local or global. Local explanations provide explanations to each prediction of the model while global explanations provide explanations to the global behavior of the model Molnar [2020].

2.2.4 Model Performance Vs. Interpretability

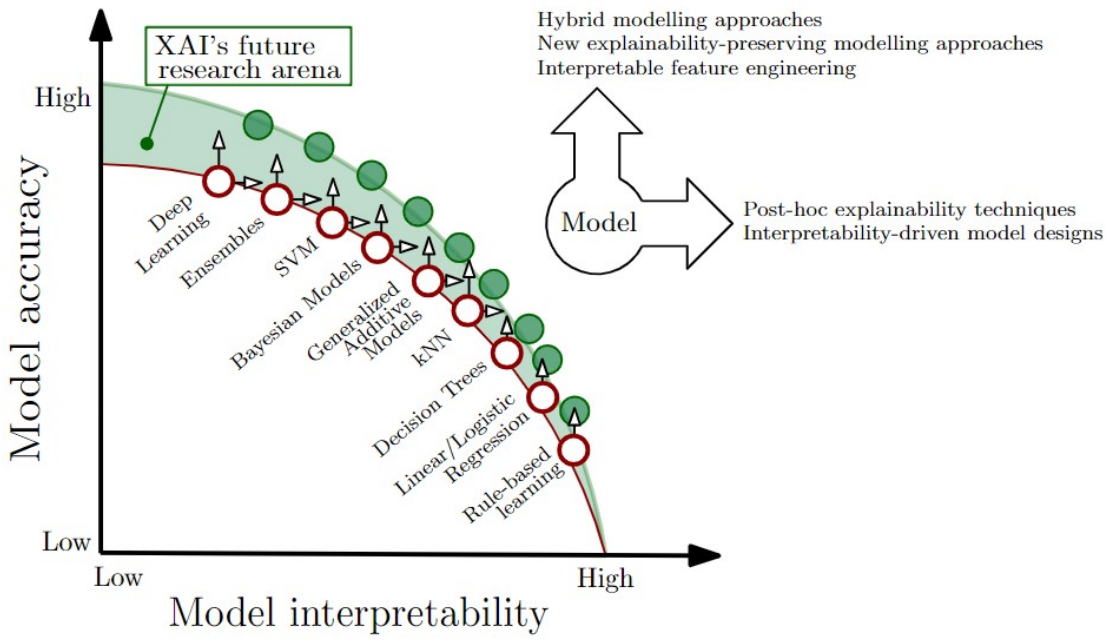


Figure 2.7: **An overview of trade-off between ML model performance given by accuracy and model interpretability**. The green shaded area denotes the potential improvements provided by research into XAI

As shown in Figure 2.7, there exists a trade-off between model performance in terms of classification accuracy and interpretability in ML/ANN models. According to Rudin [2019], complex models like

boosting-tree ensembles are more accurate but are less interpretable and comprehensible to model users. On the contrary, rule-based models and decision tree models are less accurate but easy to interpret by model users. The potential for XAI approaches lies in the green shaded area in Figure 2.7. The goal for XAI research and adoption is to improve model interpretability while not compromising on model accuracy.

2.3 Three phases of Explainable AI

According to Anjomshoae et al. [2019], three key explanation phases are crucial for the success of an XAI system - Explanation Generation, Explanation Communication, and Explanation Reception. Ehsan and Riedl [2020] argues that explainability in AI is also a human-computer interaction (HCI) problem as it is a technical problem in AI. Therefore, any adoption or development of the XAI system should involve a human-centric sociotechnical approach, rather than a purely technical approach. The three phases of an XAI system that conforms to a socio-technical approach are,

1. **Explanation Generation:** Explanation generation can be a very complex task as there are numerous XAI approaches to investigate. There are no standard ways to determine the optimal approach for an AI system depending on the explainee requirements. Hall et al. [2019],Liao et al. [2020] and, Wolf [2019] have introduced frameworks to bridge the gap between explainee requirements and the XAI approaches. Although, these frameworks have neither been practically investigated nor implemented within the financial services.
2. **Explanation Communication:** Explanation communication involves the effective communication of the explanation generated to the explainee. Explanations can be communicated textually or visually while visual communication through interactive dashboards is mostly favored Kaur et al. [2020] & Bhatt et al. [2020]. The research into novel, state-of-the-art interactive dashboards are on the rise as AI developers and end-users seek more interactivity with explanations Collaris and Van Wijk [2020]
3. **Explanation Reception:** This phase is crucial to evaluate if the explainee truly understands the explanations to AI system behavior or output. Doshi-Velez and Kim [2017] calls for a human-grounded evaluation approach to measure the quality of the explanation generated and communicated. Performance metrics for explanation goodness and explainee satisfaction are used to evaluate the quality of explanation in Hoffman et al. [2018].

3 Research Problem & Research Framework

.

3.1 Problem Context

Due to the increasing adoption of information technology by businesses and entrepreneurs, digital payment systems are widely used for conducting online financial transactions. The online financial transactions facilitated by the digital payment systems are faster than traditional transaction systems Syahadiyanti and Subriadi [2018]. The digital payment systems are developed and deployed by Dutch financial institutions to facilitate reliable and fast financial transactions for the benefit of their customers. For example, Tikkie is a mobile payments application developed by ABN AMRO Bank. Tikkie facilitates online financial transactions for a customer base of 7 million users. To no surprise, online transaction fraud has increased along with the increasing adoption of online financial transactions through mobile payment systems or credit cards. According to UK Finance, the total losses in 2019 due to financial fraud in the UK amounts to £828.8 million, and in that losses due to online transaction fraud through mobile payment systems amounts to £455.8 million Worobec [2020]. The global financial losses due to online transaction fraud should be significantly higher. Therefore, both financial institutions and regulators in the Netherlands are deeply interested in developing reliable fraud detection systems that detect and prevent online transaction fraud. Transaction fraud detection systems are autonomous systems that detect anomalous or suspicious transaction flows between two or more entities. The fraud detection systems have traditionally been rule-based models but increasingly Machine Learning (ML) models are used nowadays Kou et al. [2004] & Zhang and Trubey [2019]. Subsequently, the suspicious transactions detected by the fraud detection system are verified by a fraud detection analyst to determine if it is fraudulent or not and then proceed to report it to law enforcement authorities. Here, the fraud detection analyst depends on the rule-based system or ML models for decision support.

In this section, an in-depth explanation is given to the current situation of fraud detection systems within financial institutions in the Netherlands along with the necessity for model explainability and interpretability. In Section 3.1.1, a stakeholder map has been provided to showcase the supervisory responsibilities of the different stakeholders involved in this problem arena along with the regulatory and legal context for fraud detection and Anti-Money Laundering (AML) in the Netherlands. In Section 3.1.2 & Section 3.1.3, the current situation of fraud detection process along with their limitations are discussed and in Section 3.1.4 & Section 3.1.5, the adoption of Machine Learning (ML) in transaction fraud detection and the regulatory implications of it are discussed.

3.1.1 Stakeholder Map & Regulatory Context for Fraud Detection

In the Netherlands, the *Dutch Financial Supervision Act or Wet op het financieel toezicht (wft)* provides a legal framework for the regulation of Dutch financial institutions [1]. The act lays down the conduct of functional supervision of the financial institutions in the Netherlands. The act also decrees the establishment of financial regulators for the prudential and conduct of business supervision of Dutch financial institutions. The supervision of Dutch financial institutions is the responsibility of the financial regulators - De Netherlandse Bank (DNB) and The Netherlands Authority for the Financial Markets (AFM). DNB and AFM have also been mandated under Anti-Money Laundering and Anti-Terrorist Financing

[1] https://www.dnb.nl/en/sector-information/supervision-laws-and-regulations/laws-and-eu-regulations/financial-supervision-act/

Act or *Wet ter voorkoming van witwassen en financieren van terrorisme* (Wwft) to supervise fraud detection and anti-money laundering activities in the Dutch financial institutions. Wwft has been enacted following the EU directives on fraud detection and AML. According to WWft Article 1a(3), the financial institutions in the Netherlands that should detect and report fraud in their infrastructure are banks, payment services, investment firms, electronic money institutions, money exchange institutions, life insurers, and payment service agents[2]. As shown in Figure 3.3, there are several key stakeholders in addition to DNB and AFM, that are instrumental in imposing fraud detection and AML requirements on individuals, financial institutions and other legal entities in the Netherlands dnb [2019]. There are three key stakeholder groups in the problem arena - regulatory authorities, law enforcement authorities, and the Dutch financial institutions. The supervisory responsibilities of these stakeholders in the context of fraud detection and AML activities are given in Table 3.1 dnb [2019]. To effectively supervise Wwft compliance, detect, investigate and prosecute fraud in the Netherlands, the three key stakeholder groups should act collaboratively dnb [2019]. The Dutch financial institutions have set up transaction fraud monitoring processes within their digital infrastructure. These fraud monitoring processes help the financial institutions to stay compliant with Wwft by detecting and investigating fraud. The following section discusses the current state of transaction fraud monitoring processes within financial institutions.

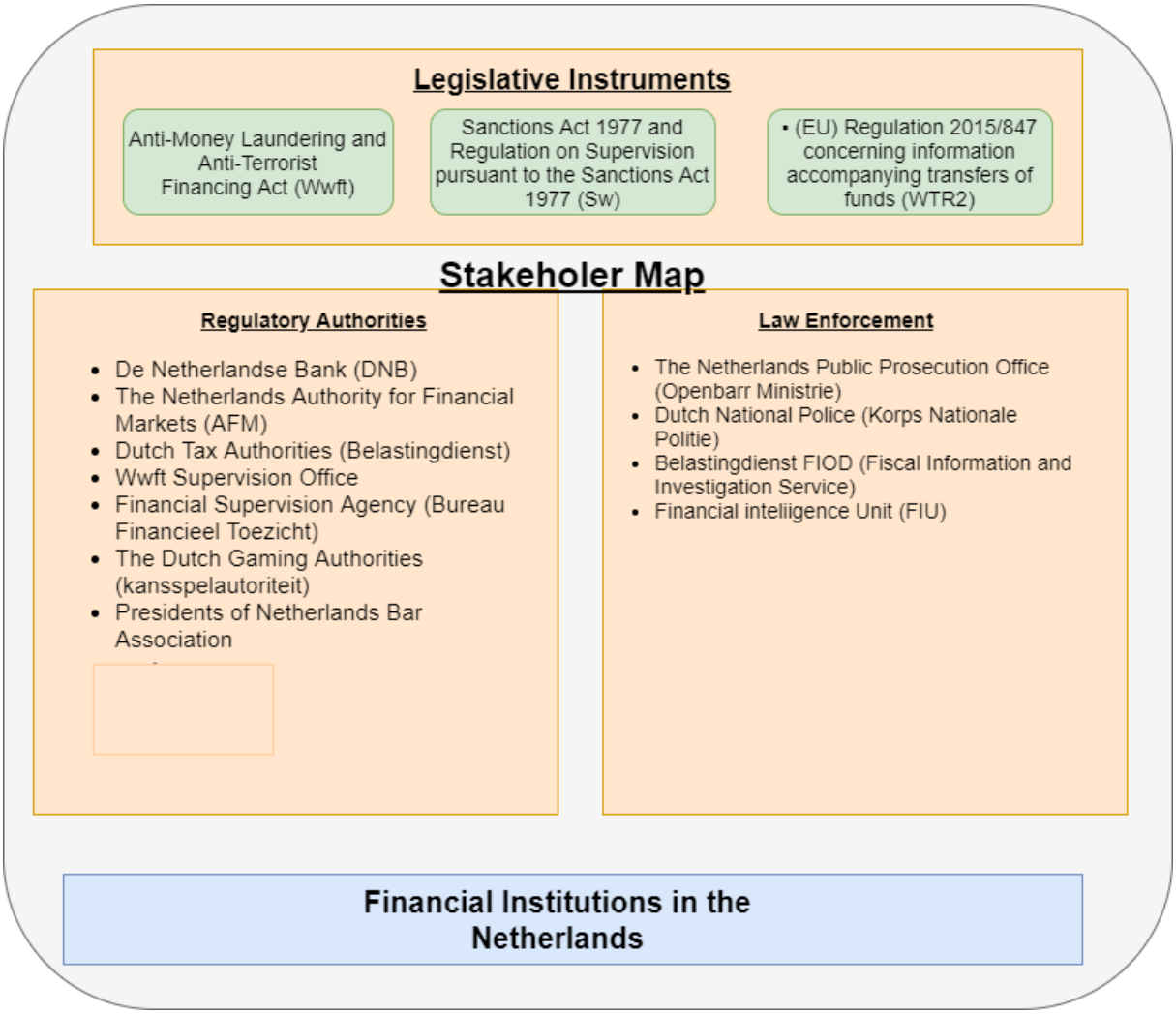


Figure 3.1: Stakeholder map of the problem arena and legislative instruments for imposing regulatory requirements on the Dutch financial institutions in the context of fraud detection and AML

[2] https://wetten.overheid.nl/BWBR0024282/2021-07-01#Hoofdstuk1

Stakeholders	Supervisory Responsibilities
De Netherlandse Bank (DNB)	DNB monitors financial institutions for compliance with Wwft. DNB enforces sanctions on non-compliant financial entities as stipulated under Wwft.
The Netherlands Authority for Financial Markets (AFM)	AFM reviews and monitors compliance with Wwft along with DNB.
Dutch Tax Authorities (Belastingdienst Anti Money Laundering Centre- AMLC)	AMLC is a knowledge and expert centre for research into fraud detection and AML tasks. AMLC coordinates with law enforcement authorities and financial institutions to monitor fraud and money laundering in the Netherlands.
Wwft Supervision Office	Enforces Wwft regulations on real-estate firms, traders, money brokers and pawnshops.
Financial Supervision Agency	Enforces Wwft regulations on tax advisers and accountants.
Dutch Gaming Authorities	Enforces Wwft regulations on casinos.
Dutch National Police	Investigates fraud, money laundering, and terrorist financing cases at the provincial and national level
The Netherlands Public Prosecution Office	Prosecutes criminals at the national level
Belastingdienst FIOD (Fiscal Information and Investigation Service)	Investigates organized crime and large-scale fraud
Financial Intelligence Unit (FIU)	FIU coordinates with financial institutions to gather information on fraudulent transactions and money flows
Financial Institutions	Client Due-Diligence - Properly monitor and investigate fraud and money laundering activities of customers in a timely manner. Know Your Client (KYC) - Gather GDPR compliant personal information on the customers' needed for investigation of fraud and money laundering.

Table 3.1: The supervisory responsibilities of the key stakeholders in the context of fraud detection and anti-money laundering activities in the Netherlands

3.1.2 Transaction Fraud Monitoring Process

The Dutch financial institutions like ABN AMRO Bank are quickly transforming into "e-banks" due to the digitization of global financial markets and the changing strategic outlook from the regulators like De Nederlandsche Bank (DNB). The "e-banks" face multitude of regulatory challenges from DNB and AFM. The toughest of the challenges are centered around financial transaction fraud monitoring Barraclough et al. [2013] and anti-money laundering (AML) Choo [2015]. Due to the lackluster performance of the banks in fraud monitoring and AML tasks, they have collectively lost $321 billion in regulatory fines Gade et al. [2019]. A large Dutch banking institution has been prosecuted for not properly monitoring fraud and money laundering through its infrastructure Toby Sterling [2021]. Fraud and money laundering weakens the global financial stability but also acts as a conduit for terrorist funding Safdari et al. [2015]. The Dutch "e-banks" need to act as gatekeepers to monitor fraud and money laundering happening through their infrastructure and prevent criminals from harming national security. Therefore, fraud monitoring and anti-money laundering are critical for the Dutch financial institutions to protect their customers' interests, safeguard national security and avoid stringent regulatory fines.

Several discussions have been made with model developers and fraud detection analysts within the Dutch financial institution sponsoring this research to understand the current situation of fraud monitoring and anti-money laundering (AML) processes. The discussions focused on the current issues with

these processes, upcoming regulatory changes, and future requirements. The discussions led to an in-depth system-level understanding of the transaction fraud monitoring processes within the financial institution. A simplified system-level process model of a typical fraud monitoring system is shown in Figure 3.2.

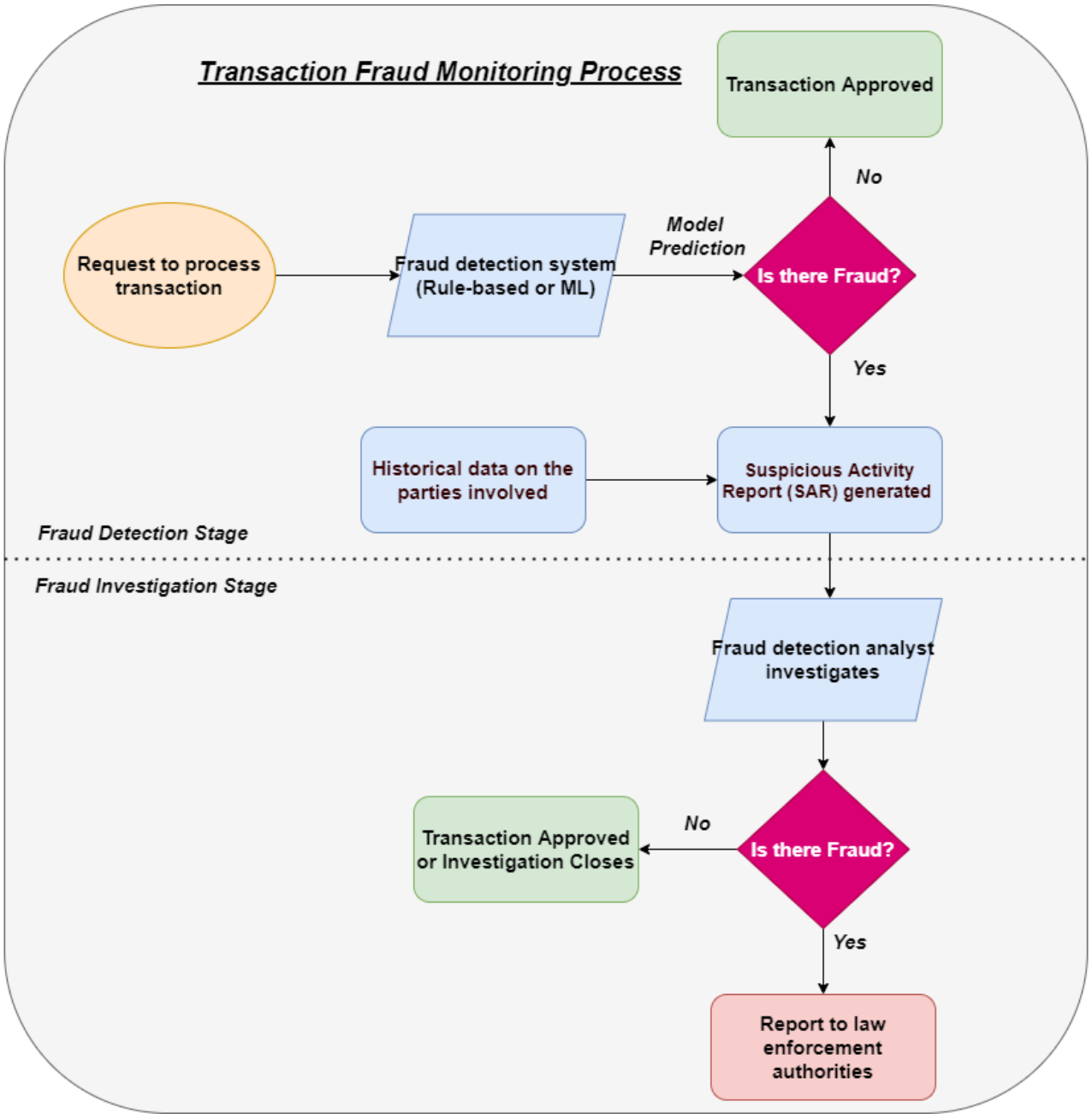


Figure 3.2: An overview of system-level process model of transaction fraud detection in the financial institution.

The fraud monitoring process involves two stages - fraud detection and investigation stages. Fraud detection systems such as rule-based or ML are used for detecting fraudulent transactions in the detection stage. Fraudulent transactions detected in the detection stage are subsequently investigated by fraud detection analysts in the investigation stage. This ensures that humans are in the loop and the system doesn't make fully autonomous decisions. Human in the loop is a requisite for fraud monitoring and AML activities as mandated by Wwft. In the case of mobile payment systems, transaction requests are voluminous Sankaran and Chakraborty [2020]. The requests are required to be processed by the bank-

ing institutions as quickly as possible for the convenience of the customers' transaction at a point of sale like in a retail store. Therefore, the fraud detection models in the detection stage of the process should be quick enough to accept genuine transactions and decline fraudulent transactions. Once a fraudulent transaction is detected to be anomalous or fraudulent, it is declined in the detection stage and forms the subject of a Suspicious Activity Report (SAR). SAR forms the basis upon which fraud detection analysts make their final assessment on the nature of the transaction - genuine or fraudulent in the investigation stage. SAR provides a comprehensive view of the circumstances of the transaction including previous transaction behavior of the parties involved Naheem [2018], Axelrod [2017]. Once the fraud detection analyst determines a fraudulent transaction has been initiated or completed, the assessment is forwarded to the law enforcement authorities like the Dutch Financial Intelligence Unit (FIU) for further investigation. The transaction fraud detection systems in the detection stage are traditionally rule-based but they suffer from several problems with performance and adaptability to new patterns of fraud. Kou et al. [2004]. The following section explores the various problems with rule-based fraud detection systems.

3.1.3 Problems with Rule-based Fraud Detection Systems

Several millions of transactions are processed in a single day by the financial institution sponsoring this research. A significant minority of these transactions is fraudulent and this makes it immensely hard to detect the fraudulent transactions Modi and Dayma [2018]. The transaction data is heavily class imbalanced with a high dimensional feature space Lopez-Rojas et al. [2016]. Due to the high-class imbalance, developing any sort of heuristic algorithm to detect fraud (minority class) becomes significantly harder. The most common types of fraud are through phishing and social engineering so it becomes hard to control fraud at the source. Usually, credit card details are phished at payment gateways and ATM kiosks, and fraudulent transactions are made using these details. Fraudsters employ social engineering to target gullible banking customers into divulging sensitive banking information and using it to commit fraud. Fraudsters keep changing their tactics and find new ways to commit fraud. Rule-based systems are static and don't recognize emerging patterns of fraud or money laundering. Rule-based systems have lack-lustre performance in detecting new patterns of fraud leading to high false-negative rate (FNR) Kou et al. [2004] & Modi and Dayma [2018].

The Dutch financial institution sponsoring this research uses rule-based systems for fraud detection. The system classifies financial transactions as genuine or fraudulent based on a specific rule-set of if-else conditionals. The system rules are pre-determined by a human expert based on the existing patterns of fraud. As the rules for classification are decided by human experts, the detection process is very inefficient against novel patterns of fraud and money laundering Watkins et al. [2003]. When the system detects a fraudulent transaction, it generates a Suspicious Activity Reports (SAR). The fraud detection analysts use this report to make a final decision if the transaction is fraudulent or not Xie et al. [2020]. There is a high risk for fraudulent transactions to be tagged as normal, leading to high FNR. Increased FNR lead to fines from the Dutch financial regulators for not being Wwft compliance dnb [2019]. There is also a risk for normal transactions to be tagged as fraudulent leading to a high false-positive rate (FPR). This leads to the loss of productive man-hours needed to analyze data associated with SAR. Increased FPR leads to increased labor costs for the financial institution. Owing to the problems with rule-based systems, ML techniques are being increasingly adopted for fraud detection Xie et al. [2020]. The following section gives reasons for the shift away from rule-based systems and towards ML.

3.1.4 Machine Learning for Fraud Detection Systems

Machine Learning (ML) techniques provide a substantial improvement over the rule-based system by lowering FNR and FPR of fraud detection systems Xie et al. [2020]. Zhang and Trubey [2019] shows that ML techniques such as Support Vector Machines (SVM), Random Forest (RF), Decision Trees (DT), and Artificial Neural Networks (ANN) improve the predictive performance of fraud detection systems. Moreover, ANN models have better predictive performance compared to linear ML models like SVM or Logistic Regression for fraud detection tasks Zhang and Trubey [2019]. Rule-based systems are static

and don't automatically change their system rules/behavior based on newly available data Kou et al. [2004]. On the contrary, ML and ANN models are dynamic and can automatically change their system behavior based on newly available data. The reason being that rule-based systems are modeled based on expert knowledge while ML models are trained using data. Therefore, ML and ANN models for fraud detection systems require less human intervention to maintain sufficient predictive performance compared to rule-based systems. The main disadvantage of ML and ANN over rule-based systems is the black-box nature of the systems. ML and ANN systems are complex and lack interpretability but provide high predictive performance Arrieta et al. [2019] (Refer Section 2.2.4). On the contrary, rule-based systems are simple and highly interpretable while providing limited predictive performance Kou et al. [2004]. Due to the lack of interpretability and transparency in ML and ANN systems, the adoption of such systems has regulatory and legal implications for the Dutch financial institutions. The regulatory and legal implications of ML and ANN-based fraud detection systems in the Dutch financial institutions are discussed in the following section.

3.1.5 Stakeholder Map & Regulatory context for Explainable AI

From the previous sections, it is clear that major disadvantages with rule-based systems are their performance and adaptability to emerging patterns of fraud. While ML and ANN-based fraud detection systems provide major improvements over performance and adaptability, they lack interpretability and transparency Arrieta et al. [2019]. There are several regulatory and legal implications for the financial institutions that use ML and ANN-based fraud detection systems that lack interpretability and transparency. A major legislative instrument that directly impacts the use of ML and ANN for fraud detection is the EU's General Data Protection Regulation (GDPR). GDPR that came into force in 2018, provides a comprehensive legal framework for protecting the data privacy rights of EU gdp [2016]. GDPR is promoted as a direct solution to the data privacy challenges faced by EU citizens due to the rapid growth of digital technologies like AI Li et al. [2019]. Article 22 of GDPR has direct regulatory and legal challenges for financial institutions that use ML/ANN-based fraud detection systems to process customers' data. Article 22 mentions that any "Automated individual decision-making, including profiling" should be provided with regulatory oversight Franklin [2019]. It provides rules that give customers the ability to chose not to be included in automated decision-making and also gives them the right to seek explanations of decisions made using their personal data gdp [2016]. Although, GDPR guarantees "right to explanation", the fact that it appears in the non-binding section of the regulation has given rise to doubts regards the actual implications of GDPR on the use of AI Wachter et al. [2017]. A key financial regulator in the Netherlands - DNB, has taken note of GDPR's "right to explanation" and provided guidelines for financial institutions regarding the development and use of AI Van der Burgt [2019]. The guidelines call for financial institutions to audit ML/ANN models and provide a detailed report regarding their inner working/model behavior to the DNB. The audit report should detail the impact of ML/ANN models on the customer's right to data privacy.

In April 2021, the European Commission published the proposal for a new EU Artificial Intelligence (AI) act. The proposal makes clear that the upcoming AI act will clarify the supervisory requirements of EU nation-states, corresponding national regulators, and financial institutions in the context of developing and using AI systems aia [2021]. The new AI act will supersede GDPR in terms of providing a legal framework for regulating the use of AI systems. The proposal defines AI as an autonomous application that provides decision-making or decision-support to humans. ML and ANN-based fraud detection systems fall under the category of AI as per the definition of the EU proposal. Therefore, the new AI act will have far-reaching consequences for the financial institutions adopting ML/ANN models for their fraud detection systems.

The EU proposes to regulate AI in the following ways through the promulgation of an AI act aia [2021],

- AI systems in the EU market should be used safely and lawfully.
- Provide legal clarity to the use of AI, to foster innovation.

- Setup regulations so that governance measures are in place so that AI systems obey fundamental rights and safety of EU subjects
- Develop a "single market for lawful, safe and trustworthy AI applications"

In addition to providing a future-proof definition of AI, the proposal clarifies the meaning of "high-risk" AI applications. High-risk AI applications can be any that "impacts the safety and fundamental rights" of EU subjects. Fraud detection systems in financial institutions protect innocent customers' economic stability. Fraud detection systems protect society by working against harmful criminals who launder money to fund mafias, terrorism, or illegitimate businesses. Fraud detection systems have a direct impact on the "safety and fundamental rights of EU subjects" aia [2021] and therefore, they are considered high-risk applications. In article 13, the proposal mandates that high-risk AI applications should be developed to be transparent and their model outputs interpretable by model users. In article 14, the proposal mandates that model developers should use "interpretation" tools and methods for enhancing the interpretability of the high-risk models. In the scientific literature, XAI approaches are seen as powerful "interpretation" tools for enhancing explainability or interpretability of the high-risk AI systems Arrieta et al. [2019] & Gunning et al. [2019].

3.1.6 Main Research Objective

As ML models provide a significant performance improvement over the rule-based fraud detection systems, their adoption in the Dutch financial services has become inevitable. Similarly, the overall adoption of AI in the Dutch financial services has been widespread Van der Burgt [2019]. The proposal for a new EU AI act signifies that an EU-level legal framework for the regulation of AI is around the corner aia [2021]. Considering the lack of existing AI regulations and to bridge the gap with upcoming regulations, the DNB provided a set of guiding principles for the responsible adoption of AI within the Dutch financial institutions. The guiding principle for responsible adoption of AI is called the "*SAFEST*" framework. As seen below, one of the guiding principles of "*SAFEST*" framework wants model developers to focus on the "explainability, simplicity, and reliability" of AI systems in addition to system performance. The

"The criteria for model choices include considerations other than quantitative evaluation metrics, such as the explainability, simplicity, and reliability of the chosen models."

The following are the six guiding principles from the "*SAFEST*" framework for the adoption of responsible AI systems Van der Burgt [2019],

- **Soundness:** The DNB emphasizes that ML/ANN-based applications should be 'compliant-by-design' and it should be "reliable, accurate and safe". The Dutch financial institutions should design ML-driven solutions within the limits of financial and non-financial regulations like GDPR. Soundness is of primary concern for DNB as it asserts that ML-model risk mitigation and strict compliance to regulations are necessary for the continued growth of ML-based applications in financial services.
- **Accountability:** The DNB believes that due to the black-box nature of ML models, it is inevitable that damages occur to customers. Therefore, the Dutch financial institutions should operationalize accountability and responsibility for ML-based applications. The DNB emphasizes that "model complexity" or the black-box nature of the ML models shouldn't be used as an excuse to evade accountability. Therefore, the Dutch financial institutions should integrate accountability for any ML-based application in their risk management strategies.
- **Fairness:** Fairness is crucial for improving individual or collective trust in the financial institutions and more specifically on the ML-driven applications. Therefore, the DNB asks the financial institutions to "define and operationalize" fairness metrics for their ML-based applications.

Figure 3.3: **An overview of the stakeholders and policy instruments impacting the regulation of AI in the Netherlands** - DNB has prudential supervisory authority over the Dutch financial institutions in the context of AI adoption while AFM has conduct of business supervision. VNB is an association of Dutch banks, that acts as a bridge between the banks and the government over the interests of the public. Dutch Data Protection Authority has supervisory authority over the use of personal data and monitors GDPR compliance regarding automated decision-making.

- **Ethics:** The DNB wants financial institutions in the Netherlands to not only comply with regulations but also fulfill moral and ethical obligations with their clients. For this, the DNB requests the financial institutions to standardize ML-model-related ethical code in their enterprise ethical policies.

- **Skills:** The DNB makes it clear that senior risk management executives within the financial institutions have to be trained on ML-based decision making. Ensuring proper training and skill development can help the management to identify ML-model-related damages and operationalize accountability for the damages.

- **Transparency:** Transparency in ML-based applications means that the financial institutions should be able to "explain" in detail the functioning of these applications. It is important to incorporate the "traceability and explainability" of ML-model decision-making in the business model.

In addition to the above high-level system requirements, there are several low-level model requirements that are specific to the use-case at hand. ML/ANN-based fraud detection systems are essentially binary classification models that classify transactions to be - genuine or fraudulent. Siebert et al. [2020] provides classification-related model quality requirements that can be attributed to the use-case at hand and they are as follows,

- **Appropriateness**: refers to degree to which a ML model is appropriate for the use-case at hand.

- **Interpretability**: refers to the extend to which a ML model is comprehensible or interpretable to the explainee.

- **Goodness of Fit**: refers to degree of goodness of ML model fit to the data or the ability of the model to accurately approximate the decision boundary embedded in the data. Model performance evaluation metrics such as Precision, Recall and Confusion matrix can quantify goodness of fit for a Ml model.

- **Robustness**: refers to ability of a ML model to make accurate predictions while handling noisy data or data with errors and missing values.

- **Stability**: refers to the ability of a trained ML model to provide generalizable results when encountering new data.

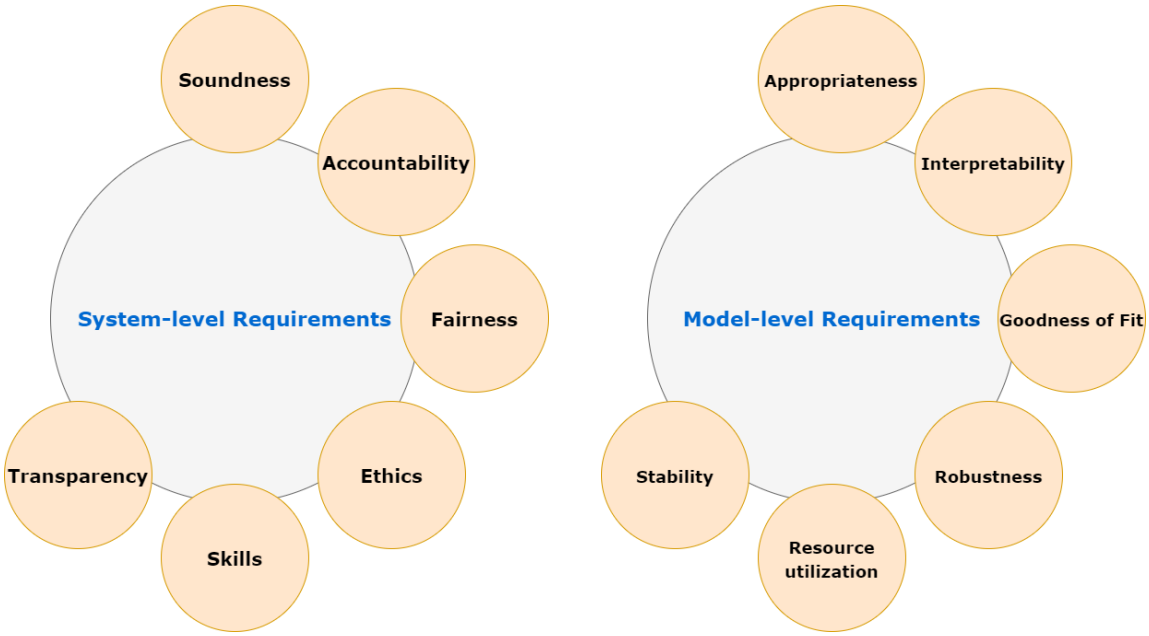
- **Computational Efficiency**: refers to the training and execution efficiency of a ML model. Efficiency measures in time and memory used for training and model inference.

Figure 3.4 gives a graphical overview of the system-level and model-level requirements for the adoption of AI in the Dutch financial institutions. The system-level requirements can be seen as a set of high-level guiding principles for the overall adoption of AI in financial services irrespective of the use case. On the other hand, model-level requirements can be seen as a set of low-level guiding principles for the adoption of ML/ANN models specific to a certain use case. The high-level system requirements can be seen as guiding principles for the higher management within the financial institutions while the low-level model requirements are for model developers. Considering the two sets of requirements along with the XAI taxonomy in Section 2.2.3, the following characteristics for an *explainable or interpretable* ML/ANN-based fraud detection model are required,

- **Binary Classification:** The ML/ANN model should be trained to perform binary classification tasks - classifying between genuine and fraudulent transactions.

- **High Class-Imbalance:** The fraud detection model should detect fraud from a highly class-imbalanced dataset - where fraudulent transactions are a significant minority of the overall dataset.

- **High Dimensional Feature Space:** The real-world transaction datasets have a large number of features. The ML/ANN model should take into consideration this high-dimensional feature space. Moreover, the model should be explainable given the high dimensionality of transaction datasets.



(a) (b)

Figure 3.4: An Overview of the Guiding Principles of AI in financial institutions at the System-level and Model level. (a) System-level Requirements per Van der Burgt [2019] (b) Model-level Requirements per Siebert et al. [2020]

- **Local & Global Explanations:** The fraud detection ML/ANN model should be locally and globally explainable. Decision-makers like fraud detection analysts should be provided with explanations for specific fraudulent transactions while model developers should be provided with global explanations for the overall behavior of the model.

- **Post-hoc Explainability Approach:** Fraud detection ML/ANN models are complex black-box models due to the demand for increased classification performance Xie et al. [2020]. As discussed in Section 2.2.4, model interpretability decreases with an increase in model performance, Keeping in mind that highly performing ML models can't be inherently interpretable Arrieta et al. [2019], these models should be post-hoc explainable.

- **Model-Agnostic Approach:** To perform optimally, ML/ANN models have to be re-trained using newly available training data Wu et al. [2020]. Similarly, the pattern and characteristics of fraud keep evolving in the real world, ML models should be constantly re-trained with newer datasets to recognize new patterns of fraud. Due to this, fraud detection models require model-agnostic explainability as the models keep updating.

In conclusion, the ML/ANN-based fraud detection systems developed within the financial institution should be able to handle high dimensional, highly class-imbalanced datasets for binary classification tasks. It should also have explanation characteristics such as model-agnostic, post-hoc, local and global explainability to overcome the black-box or interpretability problem of ML/ANN models. As proposed by the scientific literature Adadi and Berrada [2018], Arrieta et al. [2019] & Gunning et al. [2019], XAI approaches can improve the explainability or interpretability of ML/ANN-based systems to overcome the interpretability problem. This thesis research aims to investigate this by accomplishing the following research objective,

"To investigate local and global, model-agnostic post-hoc explainability as a proof-of-concept for improving explainability or interpretability in Machine Learning and Artificial Neural Network (ANN) models used for online transaction fraud detection"

3.2 Research Framework

In the previous section, the research problem and the characteristic features of an "explainable or interpretable" ML/ANN-based fraud detection system have been proposed. Based on the research problem, the main research objective has been formulated to demonstrate proofs-of-concept to investigate XAI approaches intended to improve model explainability or interpretability of ML and ANN-based fraud detection systems.

To accomplish the main research objective, four research objectives have been formulated. The academic research into XAI approaches is extensive but the practical feasibility of XAI approaches intended to improve interpretability in ML/ANN-based fraud detection systems is limited. To demonstrate proofs-of-concept for improving model explainability or interpretability of ML/ANN-based fraud detection systems by investigating XAI approaches, it is important to identify and describe the state-of-art XAI approaches available in the scientific literature. Since the XAI scientific literature is extensive, selection criteria have been formulated to select the XAI approaches that have to be investigated for demonstrating the proofs-of-concept. For accomplishing this the following research objective has been formulated,

Research Objective 1: *Describe the state-of-art XAI approaches useful for improving the model-agnostic post-hoc explainability of Machine Learning and Artificial Neural Network (ANN)-based fraud detection models*

Research Objective 1 helps in selecting and describing the state of art XAI approaches that forms the basis of the proofs-of-concepts demonstrating. Subsequently, to demonstrate the proofs-of-concept for investigating the selected XAI approaches, ML and ANN fraud detection models have to be developed and tested for their classification performance. This is achieved through the following research objective,

Research Objective 2: *Investigate the extent to which ML and ANN-based fraud detection models trained using synthetic transaction data detect financial transaction fraud*

Upon accomplishing the research objectives 1 & 2, the selected XAI approaches along with the ML and ANN-based fraud detection models developed in Chapter 6 have been used to demonstrate proofs-of-concept by investigating the XAI approaches in Chapter 8. This is achieved through the following research objectives,

Research Objective 3: *Demonstrate a proof of concept for improving explainability or interpretability of ML-based fraud detection models using model-agnostic post-hoc explainability approaches*

Research Objective 4: *Demonstrate a proof of concept for improving explainability or interpretability of ANN-based fraud detection models using model-agnostic post-hoc explainability approaches*

Research Objective 3 deals exclusively with demonstrating a proof of concept for improving model explainability or interpretability in ML-based fraud detection models by investigating suitable XAI approaches selected after accomplishing Research Objective 1. Research Objective 4 deals exclusively with demonstrating a proof of concept for improving model explainability or interpretability in ML-based fraud detection models by investigating suitable XAI approaches. The research objectives 3 & 4 are formulated separately due to the reason that ML and ANN models have fundamentally different internal mechanisms and therefore, require the investigation of different XAI approaches.

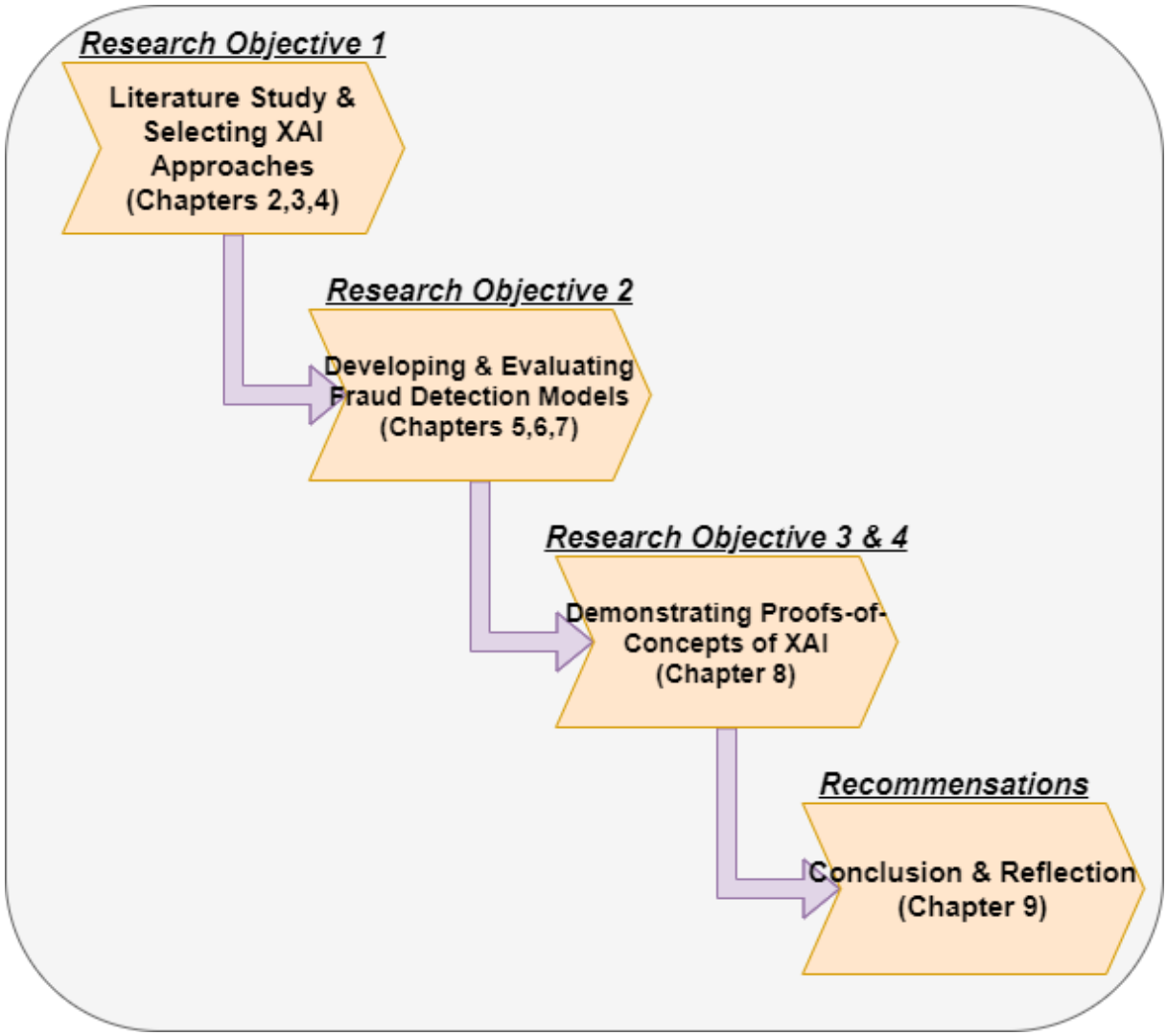


Figure 3.5: A Conceptual Overview of the Research Framework developed for this thesis research.

3.3 Research Methods

The previous section presents the research framework formulated for the thesis. In this section, the methodology employed to accomplish the four research objectives formulated in the research framework has been discussed.

3.3.1 Methodology for Research Objective 1

Describe the state-of-art XAI approaches useful for improving the model-agnostic post-hoc explainability of Machine Learning and Artificial Neural Network (ANN) based fraud detection models

Rationale for Methodological Choices

Exploratory research involving a literature study had been performed to identify and describe in detail the various model agnostic post-hoc explainability approaches relevant for the use case at hand. The

scientific literature in XAI is extensive albeit with inconsistent keywords and terminologies. Therefore, a backward snowballing approach had been performed to identify the relevant papers. Subsequently, the identified papers along with survey papers in the field of XAI such as Adadi and Berrada [2018] & Arrieta et al. [2019] had been used as primary sources for identifying key model agnostic, post-hoc explainability approaches and their corresponding literature. An XAI selection criteria (Refer 4.1) had been developed to identify and describe a set of state-of-art model agnostic, post-hoc explainability approaches in Chapter 4. The algorithmic implementations of these approaches had been presented in detail along with pseudo-code. Based on this in-depth analysis of the approaches, the advantages and disadvantages of these approaches had been understood. Based on this understanding, two state-of-art approaches had been selected for demonstrating the proofs-of-concept. This selection process in identifying suitable approaches had to be performed since there is no literature on the XAI approaches most suitable for fraud detection models trained using synthetic transaction data. Moreover, the selection process helped to find new XAI approaches in the scientific literature that had not yet been investigated for improving explainability or interpretability in ML/ANN-fraud detection models.

Data & Tools

Around 40 research papers from different domains of XAI gathered from knowledge repositories such as *IEEExplore, Elsevier Scopus, Google Scholar, and TUDelft knowledge repository*

Deliverable

A set of XAI approaches was selected from the scientific literature and described in depth along with their algorithm implementations.

3.3.2 Methodology for Research Objective 2

Investigate the extent to which ML and ANN-based fraud detection models trained using synthetic transaction data detect financial transaction fraud

Rationale for Methodological Choices

To accomplish this research objective, existing literature on the implementation of ML/ANN models for financial fraud detection had been reviewed. Before developing the ML/ANN models, the synthetic transaction dataset from PaySim had been analyzed and subsequently featured engineered to solve data quality issues. This had been done to improve the performance of the ML/ANN models trained on this dataset. Both linear (Logistic Regression & SVM) and non-linear (XGBoost, MLP) ML/ANN techniques were chosen to develop the fraud detection models. Upon training the models under various experimentation scenarios using hyperparameter optimization, SMOTE Oversampling and class-weighting, the classification performance of these models had been evaluated. Hyperparameter optimization had been performed to improve the performance of the model beyond their baseline performance. At the same time, SMOTE oversampling and class-weighting had been performed to mitigate performance issues due to the high class-imbalanced nature of the synthetic transaction datasets.

Data & Tools

The training and testing data is obtained from a synthetic transaction dataset generated using the PaySim data generator (Refer Chapter 5). Python 3.7 is used for EDA, data pre-processing, feature engineering, model training, and evaluation. Several Python libraries such as Pandas 1.2.4, Matplotlib 3.4.1, Seaborn 0.11.1, Numpy 1.20.2 are used for EDA, data pre-processing, and, feature engineering.

Scikit-learn 0.24 Python library is used to train the ML models except for the XGBoost model. The XGBoost model has been trained using Python library XGBoost 1.4.1 and hyperparameter optimization or tuning has been done using python packages Hyperopt and Ray 2.0.0. Tensorflow 2 Keras API is used for training and evaluating the MLP models. The hyperparameter tuning of the MLP models is done using Keras Tuner and HParams TensorBoard Dashboard. The model training and evaluation are performed in an HP workstation using an Intel(R) Core(TM) i7-6700HQ CPU clocked at 2.60GHz with 16 GB of RAM.

Deliverable

Training and testing of Logistic Regression, Support Vector Machines, Random Forest, XGBoost, LightGBM, and MLP models for performing fraud detection tasks. The best performing ML and ANN model among the ones developed had been chosen for demonstrating the proofs of concepts of XAI in Chapter 8.

3.3.3 Methodology for Research Objective 3 & 4

- Demonstrate a proof of concept for improving explainability or interpretability of ML-based fraud detection models using model-agnostic post-hoc explainability approaches
- Demonstrate a proof of concept for improving explainability or interpretability of ANN-based fraud detection models using model-agnostic post-hoc explainability approaches

Rationale for Methodological Choices

To accomplish this research objective, the XAI approaches identified from the scientific literature after the completion of Research Objective 2 had been investigated to demonstrate proofs of concepts for model agnostic post-hoc explainability. Feature contribution method - TreeSHAP had been investigated for improving model explainability in the best performing XGBoost-based fraud detection model. Example-based method - Diverse counterfactual had been investigated for improving model explainability in the best performing XGBoost & ANN-based fraud detection model. TreeSHAP had been investigated using only the XGBoost-based fraud detection model as it can provide higher computational efficiency by boosting tree models like XGBoost over neural networks like MLP. Diverse Counterfactuals had been investigated using both XGBoost and MLP since it is providing high computational efficiency for both the models.

Data & Tools

Three different subsets of transaction data of various transaction types had been used (Refer Section 8.1.2 for the implementation details) The tools used are python packages - TreeSHAP Explainer and DiCE for TreeSHAP and DiCE respectively.

Deliverable

TreeSHAP algorithm has been implemented on the three different XGBoost models to generate local and global explanations for model predictions using the following plots - *SHAP feature importance plots, Summary plots, and SHAP feature interaction plots*. DiCE had been used to generate various counterfactual explanations for both genuine and fraudulent transactions and visualized as Pandas DataFrame.

4 Explainable AI Approaches

In this chapter, the scientific literature on model-agnostic post-hoc explainability approaches have been identified and studied to achieve the following research objective,

Describe the state-of-art XAI approaches useful for improving the model-agnostic post-hoc explainability of Machine Learning and Artificial Neural Network (ANN)-based fraud detection models

4.1 XAI Selection Criteria

There are numerous model agnostic, post-hoc explainability approaches available in the scientific literature Adadi and Berrada [2018], Arrieta et al. [2019]. To shrink the search space and be able to find the most suitable approaches for demonstrating the proof-of-concept in the fraud detection domain, an XAI selection criteria has been formulated. The selection criteria has been used to identify model agnostic, post-hoc explainability approaches to interpret ML and ANN fraud detection models. In total 40 research papers have been reviewed using the XAI Selection Criteria detailed below. "Backward Snowballing" has been used instead of keyword search as the terminologies and definitions for XAI are continuously evolving and inconsistent. For example, survey papers such as Adadi and Berrada [2018], Arrieta et al. [2019] have been used as primary sources for identifying key model agnostic, post-hoc explainability approaches and their corresponding literature. Knowledge repositories such as *IEEExplore, Elsevier Scopus, Google Scholar,and TUDelft knowledge repository* have been used to identify the relevant scientific literature.

The following *"XAI Selection Criteria"* has been used to identify model-agnostic post-hoc the explainability approaches,

1. ***Feasibility***: The paper proposing an approach, should provide an open-source code-level implementation in a python package. The main goal of the thesis is to investigate the practicality of the approaches by demonstrating proof-of-concept and not to pursue theoretical research of the approaches.
2. ***Relevance:*** The approach should be able to provide model agnostic post-hoc explainability for either ML or ANN models.
3. ***Scientific rigor:*** The approach should have been scientifically developed and rigorously tested for various machine-learning models using either qualitative or quantitative evaluation metrics.

Figure 4.1 shows an overview of the methodological process undertaken to find the most suitable set of XAI approaches for demonstrating the proof-of-concept in chapter 8. The XAI selection criteria has been used to identify several XAI approaches under the categories of feature contribution and example-based methods.

The selected approaches are elaborated in detail with their implementation pseudo-code in the following sections 4.2 & 4.3. A deeper understanding of the code-level implementation of the approaches has helped in delineating the advantages and disadvantages of each approach to compare and contrast between the approaches (Refer section 4.4). Comparing and contrasting these approaches has helped in identifying the most suitable set of approaches for demonstrating proofs-of-concept. The methodological choices made to arrive at the most suitable set of XAI approaches have been discussed in section 4.4. By demonstrating the proofs-of-concept, the suitability or the applicability of the approaches to the fraud detection use-case has been investigated in Chapter 8.

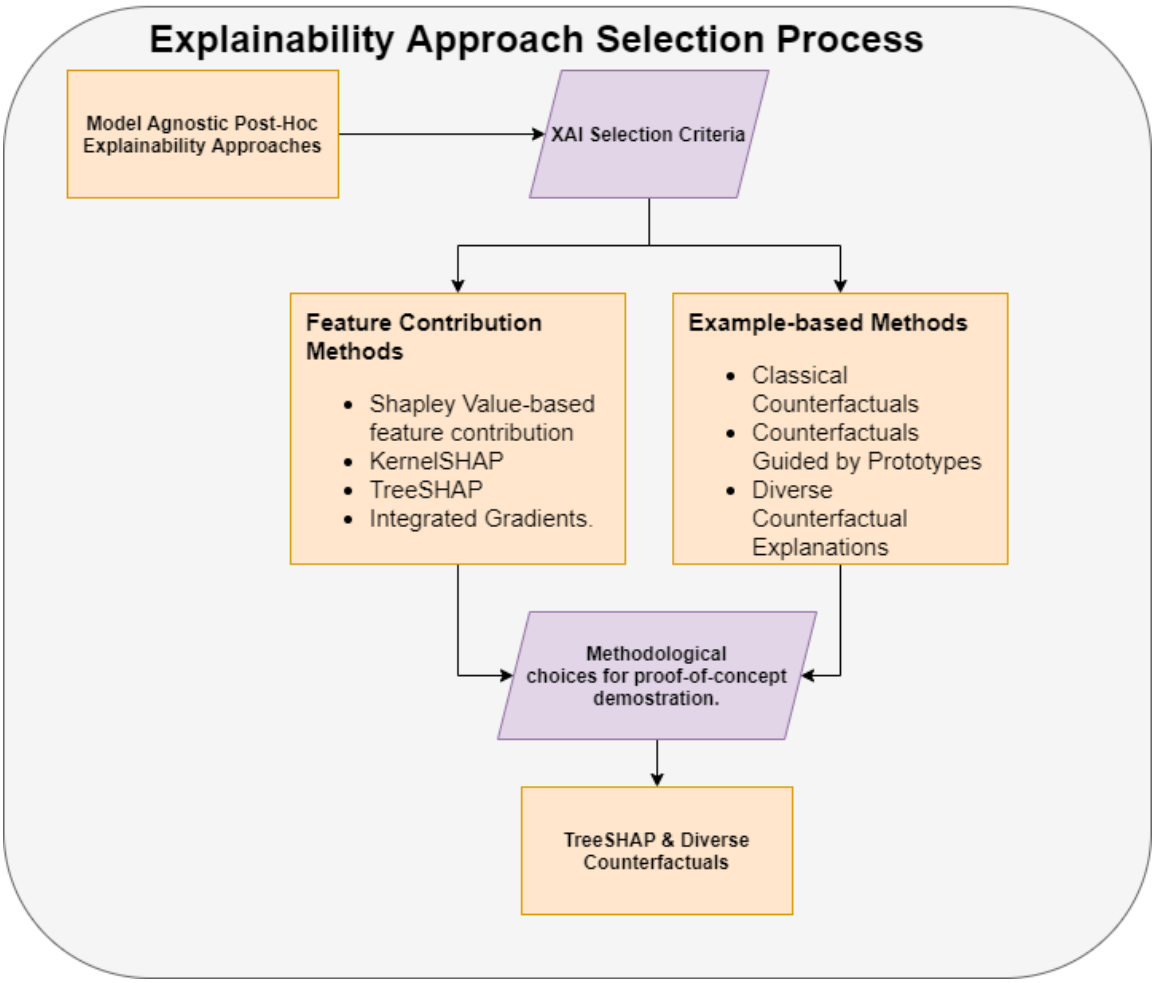


Figure 4.1: An overview of the selection process for identifying the most suitable XAI approaches for ML and ANN-based fraud detection models

Therefore in addition to the main question for this chapter,the following research sub-questions have been answered,

1. *What are the code-level implementations of the model-agnostic post-hoc explainability approaches identified through the XAI selection criteria?*
2. *What are the advantages and disadvantages of the selected explainability approaches?*

4.2 Feature Contribution-based methods

Feature contribution methods help attribute the relative importance of each input feature towards the predictions of machine-learning models. These methods provide post-hoc model explainability and interpretability based on quantifying the importance or contribution of the input features to the output Arrieta et al. [2019]. There are a large diversity of algorithms that try to rank the importance of feature values. Feature contribution methods can provide both global and local explanations. ***Permutation feature importance (PFI)*** is a post-hoc global explainability method based on feature-contribution scoring. In this method, feature permutations that lead to a break in the relationship between the feature and true outcome are used to score the prediction error increase Casalicchio et al. [2019]. The advantage of

this method is that it takes into account both individual feature effects and feature interaction effects on model performance. PFI provides highly compressed, global insights while the main disadvantages are that it needs access to the true outcomes and the results are inconsistent among each other Molnar [2020].

Shapley value based feature contribution methods provide post-hoc local explainability. Shapley value-based methods have a strong theoretical background in game theory Shapley [2016]. In section 4.2.1, Shapley value-based feature contribution method is explained in detail. ***Shapley Additive exPlanations (SHAPs)*** is another local explainability method based on Shapley values. SHAP has two implementations such as KernelSHAP and TreeSHAP where TreeSHAP is specifically for tree-based machine learning models. Both KernelSHAP and TreeSHAP are explained in detail in sections 4.2.2 and 4.2.3 respectively. In the section 4.2.4, Integrated Gradients (IG) which is a model-specific feature contribution method for ANN-models is explained in detail with implementation pseudo-code.

4.2.1 Shapley Values

Shapley value in cooperative game theory, gives the ”payouts” (predictions) that can be distributed fairly among the ”players” (features). Shapley value is given as the ”average marginal contribution of a feature value across all possible coalitions” Molnar [2020].It quantifies how big each feature contribute to the final prediction of a specific instance in comparison to the average overall prediction. Obtaining contributions to a specific instance, can help with local explainability. According to Lundberg and Lee [2017], the Shapley value for each feature is calculated by grouping the features into a ”coalition” and computing the prediction including and not including the specific feature value. The difference between the outcome including and not including the specific feature value gives the marginal contribution. The process is iterated for all possible coalition by increasing the sampling step and calculating the overall Shapley value distribution of the features for a specific prediction.

Example

For example, figure 4.2 gives the Shapley value distribution for a specific prediction of a tree-based model predicting cervical cancer using the UCI cervical cancer (Risk Factors) dataset Fernandes et al. [2017]. The figure is known as the feature importance or contribution plot. The average prediction for this dataset is 0.03, which gives the average cancer probability for women. The actual prediction for a specific woman is 0.57, which can be explained from the feature contribution plot. The ‘STDs’ feature has significantly contributed to her increased probability of cervical cancer. The difference between the average and actual prediction is given by the sum of the feature contribution values. For the sake of interpretation, Shapley values should not be assumed as the difference between the predictions when features are removed from the set of features (coalition in game theory) but as the average contribution of the features to the prediction across every possible combinations of features Molnar [2020].

Implementation

Shapley value is contributions of a feature to a prediction that is weighted and summed over all feature combinations as given by equation 4.1 Lundberg and Lee [2017]. ϕ_i gives the Shapley value of a feature i in a f model wherein S is a subset of N inputs.The Shapley value of a feature is estimated by evaluating all feature coalitions with or without the i-th feature. Computational complexity increases exponentially as the feature space of the model increases, therefore Štrumbelj and Kononenko [2014] proposes a Monte-Carlo sampling approximation method as shown in equation 4.2.

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{(|S|!(M - |S| - 1)!)}{M!} [f(S \cup \{i\}) - f(S)] \tag{4.1}$$

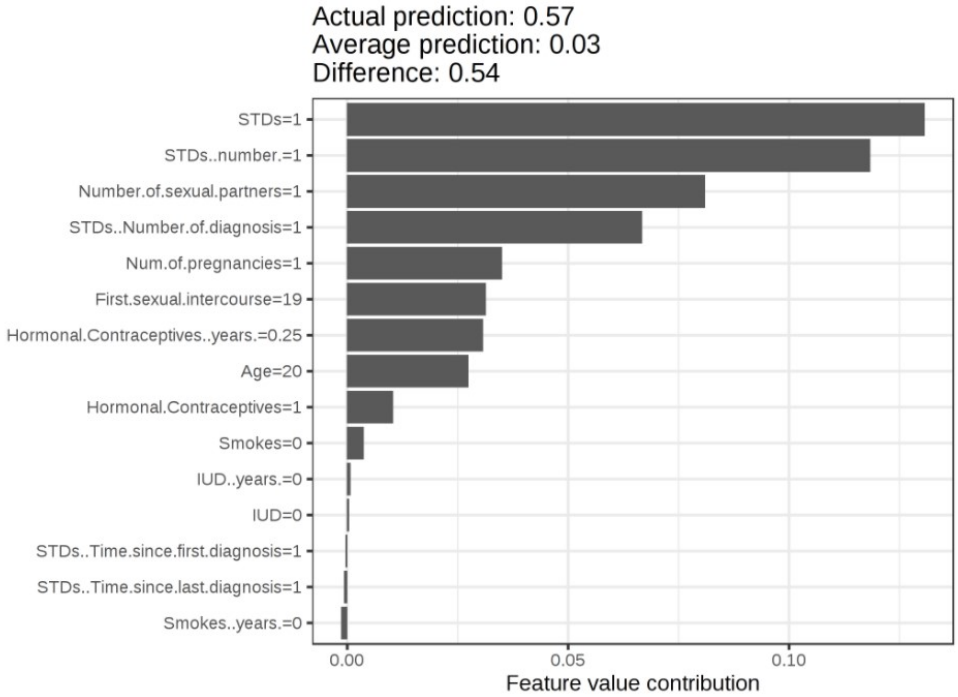


Figure 4.2: Feature contribution plot for a cervical cancer ML prediction model using Shapley values Molnar [2020]

$$\Phi_i = \frac{1}{N} \sum_{n=1}^{n} \left(f(x^n_{+i}) - f(x^n_{-i}) \right) \tag{4.2}$$

where $f(x^n_{+i})$ gives the predicted value for an instance x in a dataset X with feature index i. The values of random subset S of features are removed to place values of a randomised data point z from the data matrix X except for value of feature i.The average of the marginal contributions over N iterations gives the Shapley value of i-th feature. An overview of this implementation is shown in Algorithm 4.1 based on the description from Štrumbelj and Kononenko [2014] Molnar [2020].

Algorithm 4.1: Estimating Shapley value of i-th feature using Monte-Carlo sampling-based approximation method. Štrumbelj and Kononenko [2014] Molnar [2020]

Output: Shapley value for the contribution of i-th feature to final prediction.

for $\forall n = 1$ *to* N **do**
- *Select a random instance z from the data matrix X*
- *Select a random permutation O of features*
- *Instantiate x as* $x_O = (x_1, ..., x_i, ..., x_M)$
- *Instantiate z as* $z_O = (z_1, ..., z_i, ..., z_M)$
- *Create two new instances such as, with feature i:* $x_{+i} = (x_1, ..., x_{i-1}, ..., x_i, z_{i+1}, ..., z_M)$ *& without feature i;* $x_{+i} = (x_1, ..., x_{i-1}, ..., z_i, z_{i+1}, ..., z_M)$
- *Find marginal contribution using:* $\phi^n_i = f(x_{+i}) - f(x_{-i})$

end

Compute Shapley value using: $\phi_i(x) = \frac{1}{N} \sum_{n=1}^{N} \phi^n_i$

The advantages of the Shapley value-based explanation method is that it has a solid theoretical background as it borrows the concept of Shapley values from the well established cooperative game theory. Therefore, it is suitable for sensitive high-risk AI use-cases. Another advantage is that the Shapley

values can be used to provide contrastive explanations by contrasting the actual prediction to average prediction of a subset or the entire dataset Molnar [2020]. The disadvantages of using Shapley values are that they are computationally expensive since they always use all of the features to estimate marginal feature contribution Molnar [2020].

4.2.2 SHapley Additive exPlanations (SHAP)

"SHapley Additive exPlanations" (SHAP) is a feature contribution or feature attribution method proposed by Lundberg and Lee [2017] as an improvement over the Shapley value-based explanation methods. SHAP achieves better performance both in terms of computational efficiency and interpretability compared to Shapley value-based methods. SHAP improves upon Shapley value-based explanation methods by representing them as a linear model. This method represents the explanator g as follows,

$$g(z^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i^{'} \tag{4.3}$$

where $z^{'} \in \{0,1\}^M$ Lundberg and Lee [2017] denotes if a single feature or a set of features are observed ($z_i^{'} = 1$) or not observed ($z_i^{'} = 0$), $\phi_i \in \mathbb{R}$ (from equation 4.1) denotes the feature attribution/contribution (Shapley values) of the i-th feature to the prediction. For any X, when all the features are observed

$$z_i^{'} = 1$$

then the equation 4.3 simplifies to,

$$g(z^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i \tag{4.4}$$

As proven by Lundberg and Lee [2017], additive feature attribution methods offer three advantageous properties such as "local accuracy, missingness and consistency". Local accuracy denotes that aggregate of feature contributions always equal to prediction of the model (f) to be explained. Missingness property makes sure that the unobserved features ($z_i^{'} = 0$) in the coalition vector $z^{'} \in \{0,1\}^M$ don't contribute to the overall feature importance. Consistency property denotes that even if the model alters in a way that the marginal feature contribution of an i-th feature increases, the feature attribution (Shapley values) towards the prediction should not decrease.

Implementation

KernelSHAP is the name given to a code-level implementation of the SHAP approach.KernelSHAP is a kernel-based estimation approach that determines the feature contribution of x to the prediction.The

implementation of KernelSHAP as proposed by Lundberg and Lee [2017] is as follows,

Algorithm 4.2: Implementation algorithm of KernelSHAP Lundberg and Lee [2017]

Data: Training data X
Input : Model f and instance of interest x from X
Output: Shapley values ϕ_i
Result: Explainer g

for $k \in \{1, ..., K\}$ **do**
  Create coalitions sampled as $z^{'} \in \{0,1\}^M$
  Use mapping function $(h_x(z_k^{'})) = z$ *that converts* $z_k^{'}$ *to original feature space in X depending on if* $z_k = 1$ *or* $z_k = 0$
  Calculate predictions from the model $f(h_x(z_k^{'}))$
  Compute weights for each z_k *with SHAP kernel* $\Pi_x(z^{'}) = \frac{(M-1)}{(\binom{M}{|z^{'}|})|z^{'}|(M-|z^{'}|)}$
  Train a linear regression model $(g(z^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i^{'})$*using the weights computed using the SHAP kernel with the help of the loss function L where* $L(f, g, \pi_x) = \sum_{z^{'} \in Z}[f(h_x(z^{'})) - g(z^{'})]^2 \pi_x(z^{'})$
end
return *the coefficients of the weighted linear model* ϕ_k *(Shapley values)*

Example

KernelSHAP is an additive feature contribution method that can provide both local and global explainability. The average of the SHAP values are calculated feature-wise across the dataset to plot a global feature importance plot. A global feature importance plot can provide insights into the features that are "important" or "influential" for the global behavior of the model. For example, figure 4.3 gives the average of feature-wise SHAP value magnitudes of an XGBoost model trained to classify the annual income level of an individual. The XGBoost model is trained using the UCI census adult income dataset Dua and Graff [2017]. It is evident from figure 4.3 that the "relationship" and "age" features have high influence on the global behavior of the income classification model.

.

4.2.3 TreeSHAP

TreeSHAP is an improved variant of KernalSHAP developed to provide local and global explanations for tree-based machine learning models like gradient boosted trees (XGBoost), random forests, and decision trees. TreeSHAP reduces the computational complexity of calculating Shapley values for tree-based models from exponential to polynomial time.TreeSHAP is preferred over KernelSHAP as it can reduce the computational time for SHAP values. TreeSHAP is an additive feature attribution method as it results in the calculation of SHAP values for tree-based models and therefore satisfies local accuracy, missingness, and consistency Lundberg et al. [2019], Lundberg and Lee [2017]. In general, Shapley values can be calculated using any function but in TreeSHAP, they are only calculated using conditional expectations. TreeSHAP calculates SHAP values that is the average of the change in the conditional expectation of model output if a feature is selected over all possible feature coalitions. SHAP values are similar to the Shapley values where the only difference is that it is calculated using conditional expectations as the set function. Due to this, TreeSHAP reduces computational complexity from $O(TLM^2)$ to $O(TLD^2)$, where D denotes the maximum depth of any tree in the ensemble,L is the maximum number of leaves in any tree and T is the number of trees in the ensemble Lundberg et al. [2019]. TreeSHAP is as accurate as KernalSHAP in calculating SHAP values and performs better in-terms of computational efficiency .

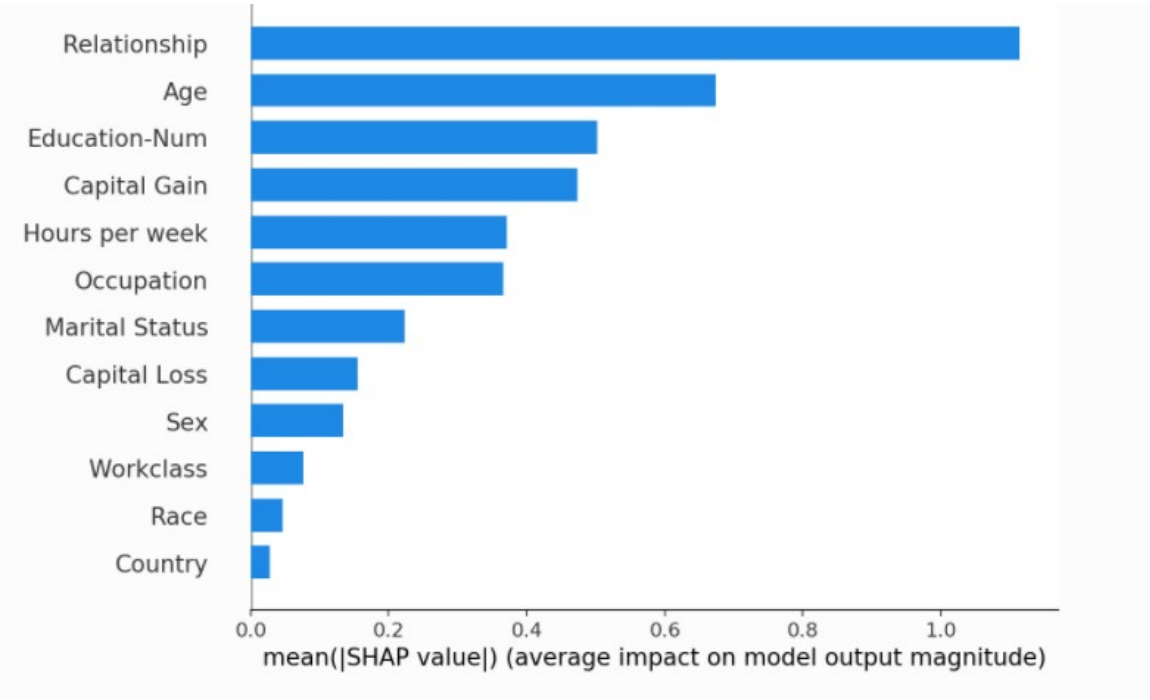


Figure 4.3: Global Importance plot of an XGBoost-based income classification model

Implementation of TreeSHAP in $O(TLM^2)$ time

Let us define $f_x(S) = f(h_x(z^{'})) = E[f(x)|x_s]$. f is the model, $h_x(z^{'})$ is the mapping function that links $z^{'} \in \{0,1\}^M$ to the original feature space, S is a non-empty set of observations in $z^{'}$. $E[f(x)|x_s]$ gives the "conditional expectation" of the output condition of $S \subseteq N$ where N is a feature input set Lundberg et al. [2019]. There are two algorithms for calculating $E[f(x)|x_s]$ in exponential ($O(TLM^2)$)or low-order polynomial time ($O(TLD^2)$).

In exponential time, we can calculate SHAP for tree-based models by computing $E[f(x)|x_s]$ and with equation 1 where $f_x(S) = E[f(x)|x_s]$ Lundberg et al. [2019]. In algorithm 4.3, the variable tree contains information about the tree such as vector v denoting the "internal node" values where a and b vectors denote the left and right. d is the vector of features indexes used for split criterion at internal nodes, vector t denotes the threshold for each node index and vector r denotes the number of data points fall in a sub-tree Molnar [2020].

Algorithm 4.3: Estimating $E[f(x)|x_s]$ - TreeSHAP in $O(TLM^2)$ time Lundberg et al. [2019]

Procedure EXPVALUE(X,S,$tree = \{v, a, b, t, r, d\}$)
  Procedure $G(j)$
    if $v_i \neq interval$ **then**
      return v_i
    else if $d_i \in S$ **then**
      return $G(a_i)$ **if** $x_{d_i} \leq t_i$ **else** $G(b_i)$
    else
      return $\frac{\left[G(a_i) * r_{a_i} + G(b_i) * r_{b_i}\right]}{r_i}$
    end
  return $G(1)$

Example

For example, figure 4.4 gives the relationship between feature values and feature contribution for a tree-based model predicting cervical cancer using the UCI cervical cancer (Risk Factors) dataset Fernandes et al. [2017]. The figure is known as a summary plot. The Y-axis gives the feature contributions to the model output and the X-axis gives SHAP value for each instance. The SHAP value of various instances that overlap at jittered along the Y-axis. This shows how much the SHAP values vary per feature. For high values of "STDs..number" feature, the SHAP value is higher. According to the model, the risk of cervical cancer increases with increase in STDs.

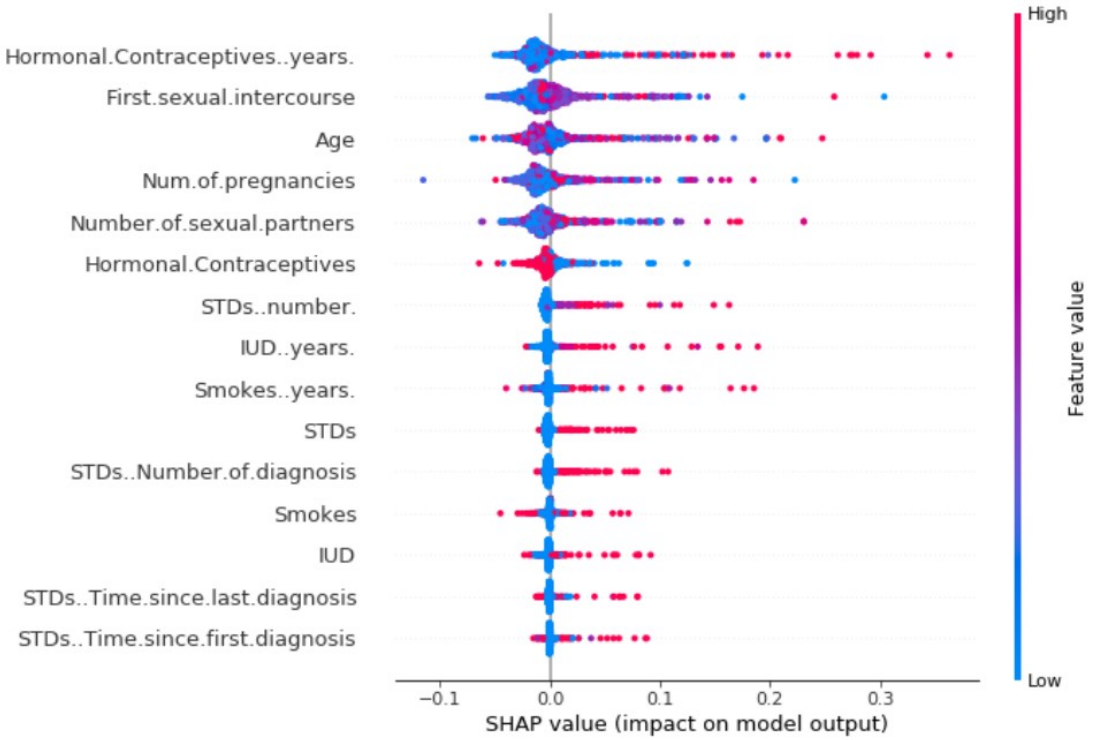


Figure 4.4: Summary plots using the SHAP values from the TreeSHAP algorithm shows the relationship between the range of feature values and feature contribution scores Molnar [2020]

4.2.4 Integrated Gradients

Integrated gradients (IG) is a model-specific feature attribution method for deep learning networks proposed by Sundararajan et al. [2017]. Feature attribution for deep learning networks refers to the distribution of "blame" or "importance" for a prediction to the different input features. Attribution can also be assigned to features in the dense layers of the deep learning network (Layer-wise attribution). Integrated gradients establish a clear link between the input and the output of any differentiable gradient-based model, to help improve interpretability. It can help ML developers debug the network and improve its capabilities. It can help end-users understand the network's strengths and weaknesses to improve decision-making capabilities. For example, a doctor using a deep learning-based image diagnosis tool for cancer treatment can understand when and how the network recognizes a tumor. Furthermore, IG can extract insights from a deep learning network that can be used to design a simpler rule-based model. IG is easy to implement, computationally efficient, and has a strong theoretical foundation.

Let us consider an input instance x and a baseline instance $x\prime$. A baseline in a deep learning network exists in an input space when the prediction in the output space is neutral. For example, a black image input to an image classification models returns a neutral prediction. The deep learning network models a function $F : R^n \rightarrow [0,1]$ where $x = \{x_1, ..., x_n\} \in R^n$ denotes the input space and $[0,1] \in Y$ denotes the output space. For a model output that is a vector, the function becomes $F_k(x)$ where k indexes the output. The attribution $A_i(x, x^{'})$ of the prediction for each feature x_i relative to the baseline input for that feature $x_{i\prime}$ is given by equation 4.5.

$$A_i(x, x^{'}) = (x_i - x_i^{'}) * \int_{\alpha=0}^{\alpha=1} \frac{\partial F(x^{'} + \alpha(x - x^{'}))}{\partial x_i} d\alpha \tag{4.5}$$

The integrated gradients as given by equation 4.5 are defined as the straight-line path integral of the between input x and the baseline input $x^{'}$ Sundararajan et al. [2017] as shown by figure 4.5.

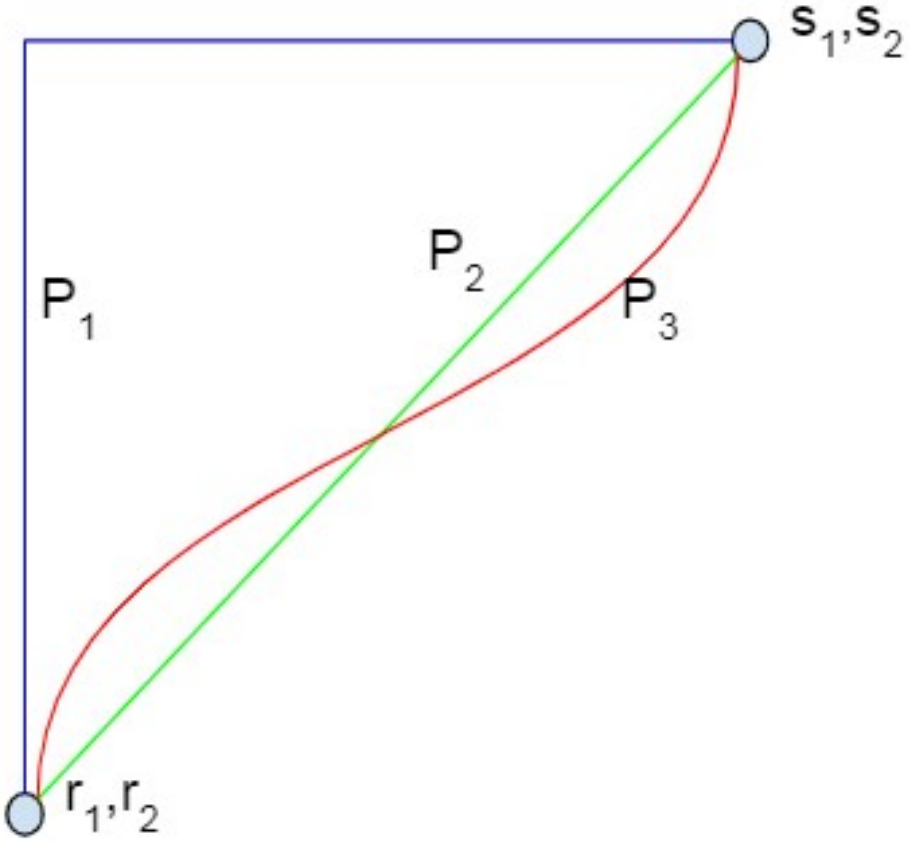


Figure 4.5: Integrated Gradients calculates the straight-line path integral P_2 between baselines (r_1, r_2) and inputs (S_1, S_2)

Integrated gradients satisfies two key axioms necessary for any attribution method - Sensitivity and Implementation Invariance.

- **Sensitivity**: This axiom states that if the input changes from a baseline $x^{'}$ to x for an i-th feature x_i and the prediction changes, then there should be a non-zero attribution assigned to x_i.
- **Implementation invariance**: The axiom states that for any two functionally equivalent deep learning networks, the attributions should be identical irrespective of the implementation.

- **Completeness**: This axiom compliments the sensitivity axiom. It states that the sum of all feature attributions across x_i must be equal to the difference between output at x and $x\prime$ as shown in equation 4.6 Sundararajan et al. [2017].

$$\sum_{i=1}^{n} A_i(x, x^{'}) = F(x) - F(x^{'}) \tag{4.6}$$

Integrated gradients are easily implemented for Tensorflow/Keras deep learning models using the *ALIBI* python package or *interpolated_path(x)* function in Tensorflow 2. The implementation algorithm of IG in ALIBI is given in algorithm 4.4.

Algorithm 4.4: Implementation algorithm for Integrated Gradients as per Sundararajan et al. [2017]

Input: Instance to be explained x, baseline input $x^{'}$, output tensor, step-size m.
Output: Visualizing the feature attributes.
Identify the input and output tensors. The output tensor is the softmax activation in a image classification network or class probabilities (logit tensor) in classification using structured data.
Select a good baseline $x^{'}$.
Select the "number of steps" m hyperparameter for the Riemman approximation of the integral.m is usually between 20 and 300 inputs as recommended by Sundararajan et al. [2017].
for $k = 1, ..., m$ **do**
    $A_i^{approx.}(x, x^{'}) ::= (x_i - x_i^{'}) * \sum_{k=1}^{m} \frac{\partial F(x^{'} + \frac{k}{m}(x - x^{'}))}{\partial x_i} * \frac{1}{m}$
end
Hyperparameter tuning: Check if sum of attributions are approximately equal to the difference between the output at x and $x^{'}$ (Check $\sum_{i=1}^{n} A_i(x, x^{'}) = F(x) - F(x^{'})$), if not increase m.
return *Approximate attributions* $A_i^{approx.}(x, x^{'})$ *for the model prediction.*
Visualize attributions as a overlay mask in case of images or histograms in class of structured data.

4.2.5 Example

Figure 4.6 gives the Integrated Gradients attribution mask overlay (Purple) for an original image of a water fountain. The ANN model is an image classification model that classifies different images based on their features. The attribution mask (purple) gives the specific pixels within the image that increased the probability of the original image to be classified as a water fountain.

4.3 Example-based methods

Example-based methods are model-agnostic, post-hoc explainability methods that use examples (instances) within the dataset to explain the data distribution or the model behavior Molnar [2020]. Unlike the feature contribution methods explained in section 4.2, the example-based method doesn't score feature values but identifies important instances. Example-based methods provide explanations understandable to non-technical stakeholders, unlike the feature contribution methods.

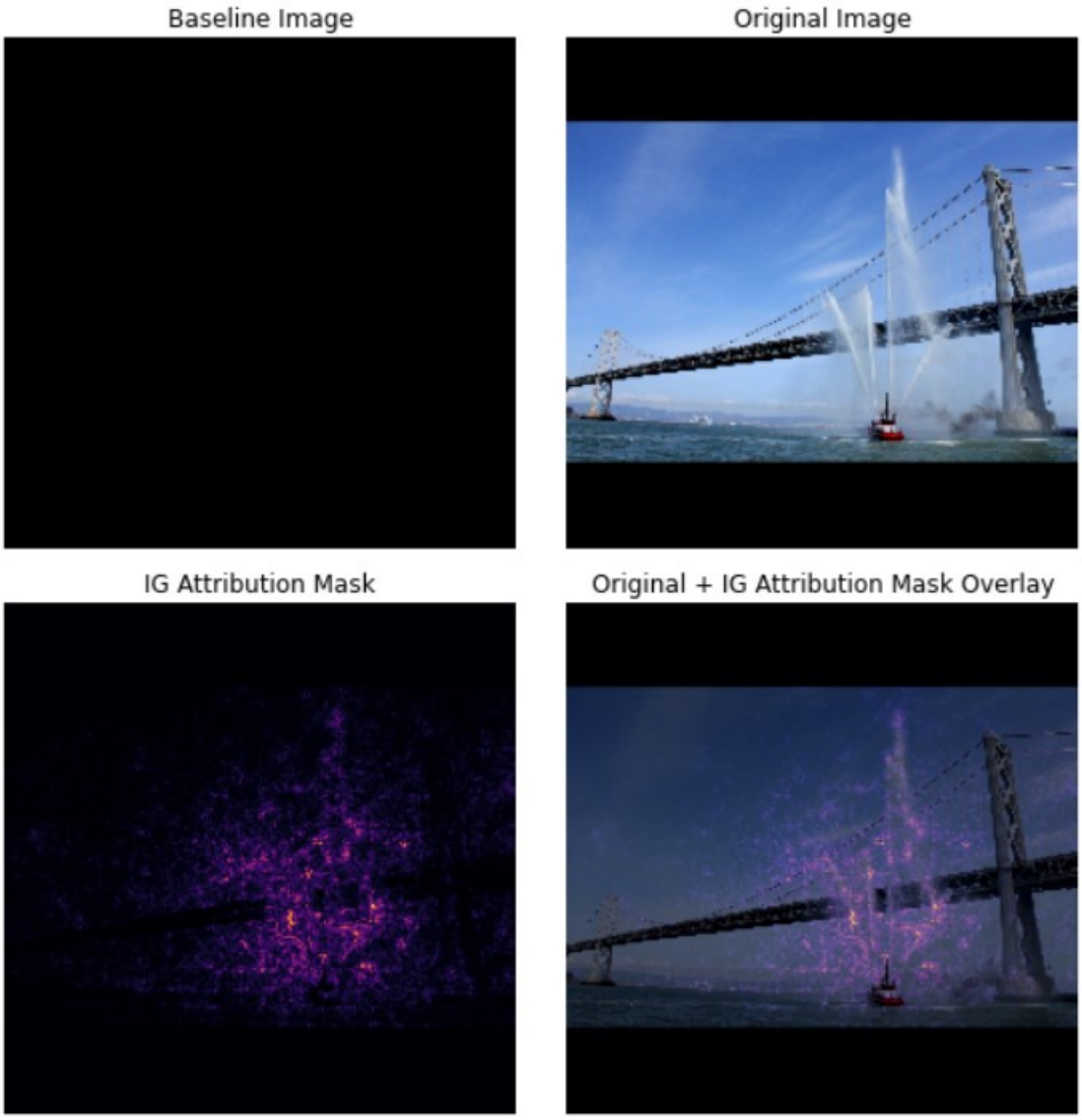


Figure 4.6: IG attribution mask overlay of an image igw [2021]

4.3.1 Classical Counterfactuals

A simple but computationally inefficient method to generate counterfactuals to an instance of interest is through "trial and error". In this method, feature values are randomly perturbed until the prediction switches to the desired outcome. A computationally efficient method is to minimize a loss function using some optimization algorithm to generate counterfactual examples. A similar loss minimization method as proposed by Wachter et al. [2018] seeks to generate counterfactual examples by minimizing the loss function at equation 4.7, where f_w is any machine learning model that is trained by tuning weights w.

$$L(x_i, x^{'}, y^{'}, \lambda) = \lambda(f_w(x^{'}) - y^{'})^2 + d(x_i, x^{'}) \tag{4.7}$$

1. The first term in the loss function L denotes the quadratic distance between the prediction for the counterfactual feature inputs $f_w(x^{'})$ Molnar [2020]and the pre-defined desired outcome $y^{'}$.

2. The second term denotes the "distance d between the instance of interest x_i and the counterfactual instance". $x^{'}$Molnar [2020].

The distance d denotes the Manhattan distance weighted feature-wise using the inverse median absolute deviation (MAD) Molnar [2020] as shown in equation 4.8.

$$d(x_i, x^{'}) = \sum_{k=1}^{p} \frac{\left| x_{i,k} - x_k^{'} \right|}{MAD_k} \tag{4.8}$$

The MAD of i-th feature over the dataset N is given by equation 4.9. The absolute distance between counterfactual and instance of interest is weighted over the inverse MAD to have all the features at the same scale.

$$MAD_k = median_{j \in N}(\left| x_{j,k} - median_{l \in N}(x_{l,k}) \right|) \tag{4.9}$$

The loss function L is solved for a parameter λ which equalises the distance between the first term (prediction) and the second term (feature values) to output $x^{'}$. A high value of λ denotes that the prediction of the counterfactual will be closer to the pre-defined desired outcome $y^{'}$ and a low value of λ denotes that the feature values of counterfactuals $x^{'}$ are closer to instance of interest x_i. Wachter et al. [2018] suggests to use a tolerance value ϵ instead of tuning λ for determining the distance between the counterfactual prediction and desired prediction $y^{'}$. Equation 4.10 gives the tolerance constraint for determining how close the counterfactual prediction should be from the desired outcome. Ideally it should be as close as possible but depending on the use-case, this constraint could vary.

$$\left| f(x^{'}) - y^{'}) \right| \leq \epsilon \tag{4.10}$$

The loss function can be minimized using any suitable optimization algorithms such as ADAM for machine learning models that expose gradients. By defining the instance of interest x_i,desired outcome $y^{'}$ and the tolerance value ϵ beforehand, the minimization of loss function gives the counterfactual instance $x^{'}$. The local optimum of counterfactual $x^{'}$ is obtained by increasing λ within the tolerance constraint ϵ as shown in equation 4.11. The implementation algorithm for the classical counterfactual method proposed by Wachter et al. [2018] is given in algorithm 4.5. ALIBI python package provides an open-source code-level implementation of this classical counterfactual method Klaise et al. [2020].

$$argmin_{x'} nax_{\lambda} L(x_i, x^{'}, y^{'}, \lambda) \tag{4.11}$$

Algorithm 4.5: Generating Classical Counterfactual Examples Molnar [2020]

Input: x_i,$y^{'}$,λ,ϵ
Output: $x^{'}$
Result: Counterfactual example
Select the instance to be explained x_i, desired outcome $y^{'}$, tolerance value ϵ and a low initial value of λ.
Initial counterfactual instance is randomly sampled.
while $\left|f(x^{'}) - y^{'})\right| \leq \epsilon$ **do**
    Increase λ.
    Loss optimization with the initial counterfactual as the beginning point.
    return *Find the local optimum counterfactual* $x^{'}$ *that minimizes the loss*
end
return *Repeat steps 3 & 5 for a list of counterfactual examples*

In general, counterfactual explanation methods suffer from the "Rashomon Effect" i.e multiple counterfactual examples are possible for a single instance of interestMolnar [2020]. To evaluate and select optimal counterfactual examples, a set of evaluation criteria are needed. Molnar [2020] Van Looveren and Klaise [2019] provides the following evaluation criteria for generating high-quality counterfactual examples $x^{'}$,

- **Criteria 1**:The counterfactual prediction should be as close as possible to the pre-defined desired outcome.
- **Criteria 2:**The feature values of the generated counterfactual should be as same in characteristics as possible to the instance of interest.
- **Criteria 3:**The input perturbations δ to the instance of interest x_i should be sparse i.e change as few feature values as possible.
- **Criteria 4:**The generated counterfactual examples should be diverse i.e a diverse set of counterfactuals should be generated for an instance to be explained.
- **Criteria 5:**The feature values of the generated counterfactuals should be interpretable and feasible i.e the feature values of the counterfactual should conform to the general and class-specific data distribution.
- **Criteria 6:**The counterfactual algorithm should be able to handle both continuous and categorical feature values together.
- **Criteria 7:**To handle, real-life datasets the counterfactual generation algorithms should be computationally efficient i.e converge to an output in less time.

The classical counterfactual approach by Wachter et al. [2018] is very simple to implement but it does not satisfy many evaluation criteria as mentioned above. The following approaches in section 4.3.2 (Counterfactuals Guided by Prototypes) and 4.3.3 (Diverse Counterfactuals) improves upon the classical counterfactual approach in generating high quality counterfactual examples.

4.3.2 Counterfactuals Guided by Prototypes

Van Looveren and Klaise [2019] proposes an improved, faster model agnostic method to generate interpretable counterfactual examples for classification tasks by using class prototypes. A prototype is a data point that is representative of the data distribution or the inherent relationship of the features in a specific class. In this approach, class prototypes are used to guide input feature perturbations by implementing an encoder or a class-specific k-d tree. By using class prototypes to guide perturbations, the

algorithm converges faster to output and provides improved interpretability. The algorithm achieves this by adding two new error terms to the simple loss function L as shown in the below equation.

$$L = c * L_{pred} + \beta * L_1 + L_2 + L_{AE} + L_{proto} \tag{4.12}$$

The first term $c * L_{pred}$ and second term $\beta * L_1 + L_2$ is similar to the first term and second term respectively in the simple loss function in equation 5. The first loss term in equation 10, tries to find a counterfactual instance that is different to the original instance. The second loss term in equation 10 tries to penalise the difference between the desired counterfactual outcome and the perturbed instance to achieve sparsity (Criterion 3 in section 4.2.1). Even though, the first two loss terms in equation 10 achieves sparsity, they don't generate interpretable counterfactuals as the training data distribution is not taken into consideration. To make the feature perturbations more meaningful, the training data distribution is used to fit an autoencoder AE and this is optimized using the error term L_{AE}. L_{AE} (equation 13). This loss term is considered as the L_2 reconstruction error of the counterfactual instance used to optimize the autoencoder AE.

$$L_{AE} = \gamma . \|x_0 + \delta - AE(x_0 + \delta)\|_2^2 \tag{4.13}$$

L_{AE} does not necessarily improve interpretability as the algorithm can only learn the overall data distribution but not the class-specific distribution. To improve interpretability and speed up the counterfactual search, Van Looveren and Klaise [2019] proposes the use of class prototypes. In here, the class prototypes are representative of the class-specific data distributions. By including and optimizing for the loss term L_{proto}, the $L2$ (Euclidean) distance between the counterfactual and the nearest class-specific prototype is minimized. Since, the feature perturbations are "guided" towards the class prototype, the search for counterfactuals speeds up and the interpretability of meaningful perturbations improves. Algorithm 7 gives the implementation algorithm for counterfactual search using auto encoders as proposed by Van Looveren and Klaise [2019]. In addition to improving interpretability and counterfactual search speed, this method can handle both continuous and categorical data by using a pairwise distance measure (Modified Value Distance Metric $MVDM$ or Association-Based Distance Metric $ABDM$). ALIBI python package provides an open-source code-level implementation of this improved counterfactual method as proposed by Klaise et al. [2020].

Algorithm 4.6: Counterfactual search with encoded prototypes Van Looveren and Klaise [2019]

Input: AE refers to the autoencoder, ENC refers to the encoder part of AE, Training sample set $X = \{x_1, .., x_n\}$, instance of interest x_0

Data: β , θ (Required), c,κ and γ (Optional)

Result: Counterfactual example x_{cf}

Label the training sample X and instance to be explained x_0 using the function f_{pred} where $X^i \leftarrow \{x \in X | argmax f_{pred}(x) = i\}$ for each class i. Also original class is defined as $t_0 \leftarrow argmax f_{pred}(x_0)$.

Define class-wise prototypes: $proto_i \leftarrow \frac{1}{K} \sum_{k=1}^{K} ENC(x_k^i)$ for $x_k^i \in X^i$ where by increasing $\left\|ENC(x_0) - ENC(x_k^i)\right\|_2$ gives the ordering of x_k^i and the constraint $K \leq \left|X^i\right|$.

Find nearest prototype p to x_0 which should be different from t_0: $p \leftarrow argmin_{i \neq t_0} \|ENC(x_0) - proto_i\|_2$.

Optimize the loss function: $\delta^* \leftarrow argmin_{\delta \in \chi} c * L_{pred} + \beta * L_1 + L_2 + L_{AE} + L_{proto}$ where $L_{proto} = \theta * \left\|ENC(x_0 + \delta) - proto_p\right\|_2^2$.

return *locally optimal counterfactual example* $x_{cf} = x_0 + \delta^*$.

4.3.3 Diverse Counterfactual Explanations - DiCE

The method proposed by Van Looveren and Klaise [2019] satisfies every criteria for generating high quality counterfactual examples mentioned in section 4.2.1 except for diversity (criterion 4) and feasibility (criterion 5). Russell [2019] is the first paper to propose novel technical approaches to generating coherent and diverse counterfactual explanations for linear classifiers (i.e. linear/logistic regression, SVM, etc.). The paper proposes a novel counterfactual search algorithm based on mixed-integer programming. A set of constraints called "mixed polytope" combined with integer programming is used to generate counterfactuals that stay consistent and coherent with the original data structure. Mothilal et al. [2020b] improves on the work of Wachter et al. [2018] and Russell [2019] by proposing a framework for generating diverse and feasible counterfactual explanations for any differentiable machine learning classifier. This method enables the user to set up constraints and context for improving the feasibility of the generated counterfactual explanations. This method improves upon Wachter et al. [2018] by formulating an optimization problem that takes into account the trade-off between the diversity of counterfactual explanations generated and the proximity to the instance of interest. A solution to the optimization problem generates many different (diverse) counterfactual examples while considering user's domain knowledge and constraints. A causal constraint-based filtering approach is used to maintain feasibility in feature input perturbations (i.e *race* or *age* feature cannot be perturbed).

Consider a machine learning model f that is both static and differentiable with binary output. Let x be the instance of interest, for which a set of k counterfactual instances $\{c_1, c_2, .., c_k\}$ are to be obtained. Both x and k are in d-dimensional space. The set of k counterfactual examples have to be both diverse and feasible. The optimization problem should consider the trade-off between diversity and feasibility, as maximizing diversity means causing big perturbations to feature values.

Diversity: Determinantal point processes (DPP) is formulated for constructing diversity in the counterfactual search algorithm.

$$dpp\_diversity = det(K) \tag{4.14}$$

where $K_{i,j} = \frac{1}{1+dist(c_i,c_j)}$ and $dist(c_i, c_j)$ gives a distance metric between two counterfactual examples in k.

Proximity (Feasibility property 1): Proximity denotes the negative vector distance between x and counterfactual example c_i. This makes sure that the counterfactual search results in c_i that is close to x. Proximity can be represented by a distance metric like l_1 distance that is optimally weighted by user-defined feature-wise weights. The proximity of k is the mean proximity over k as shown in equation 15.

$$Proximity = -\frac{1}{k}\sum_{i=1}^{k} dist(c_i, x) \tag{4.15}$$

Sparsity (Feasibility property 2): Sparsity property demands fewer changes to the feature space to generate counterfactual examples. A counterfactual example is more feasible if a fewer number of features are changed from the original input x. This constraint is not included in the loss function as it non-convex and it is achieved after output convergence using few modifications as mentioned in section 3.3 of Mothilal et al. [2020b].

User-defined constraints (Feasibility property 3): Based on the real-world context and domain knowledge, the user can impose constraints on the feature perturbations. This is done to avoid breaking causal links in the features. For example, the *race* feature cannot to changed to generate a counterfactual example.

Based on the above-mentioned constraints and properties, an overall loss function is defined for the optimization problem as shown in equation 16 as proposed by Mothilal et al. [2020b].

$$C(x) = argmin_{(c_1,...,c_k)} \frac{1}{k} \sum_{i=1}^{k} yloss(f(c_i), y) + \frac{\lambda_1}{k} \sum_{i=1}^{k} dist(c_i, x) - \lambda_2 dpp\_diversity(c_1, ..., c_k) \quad (4.16)$$

where c_i is the counterfactual example which is a subset of k. The first and second term is identical to loss function proposed by Wachter et al. [2018] (equation 5 in section 4.2.1). The first term gives the distance between desired pre-defined outcome and the model prediction for generated counterfactual c_i. The second term gives the closeness between the number of input features (d)in x and c_i. $dpp\_diversity$ is the distance metric enforcing diversity and λ_1, λ_2 are "hyperparameters that balance the loss terms" Mothilal et al. [2020b]. c_i is initialized randomly and a gradient-based optimization process is employed to minimize the loss function. In conclusion, DiCE is a novel approach to generate diverse and feasible counterfactual examples. It seeks to satisfy all the evaluation criteria for generating high-quality counterfactuals as mentioned in section 4.2.1. A main advantage of DiCE is that in addition to being model agnostic, it does not need all the data to generate counterfactuals. This feature makes it an interesting tool for explaining models trained on sensitive datasets. DiCE python package provides an open-source code-level implementation of this novel counterfactual explanation method (https://github.com/interpretml/DiCE). The implementation of DiCE is show in algorithm 8.

Algorithm 4.7: Generating Diverse & Feasible Counterfactual Examples using DiCE

Input: x,$f(x) = y^{'}$,λ_1,λ_2
Output: ($c_i = c_1, ..., c_k$)
Result: k set of counterfactual examples
Select the instance to be explained x, desired outcome $y^{'}$ (favourable Class), low initial value of λ_1 & λ_2.
Randomly initialize the feature values of c_i.
while *step* $\leq$ 5000 *or until the* $C(x)$ *converges* **do**
    Tune hyperparameters λ_1 & λ_2.
    Loss optimization with the initial sampled counterfactual as the beginning point.
    return *locally optimal counterfactual* c_i *that minimizes the loss*
end
return *Repeat steps 3 & 5 for k set of counterfactual examples*

Causal feasibility of counterfactual examples: The DiCE python package has incorporated an addition method proposed by Mahajan et al. [2019] for using causal relationships between input features to enforce feasibility constraints. The paper proposed that any feasibility constraint should be based on causal links between features and not merely statistically driven. For this, causal proximity regularizer is used instead of the proximity distance metric proposed by Mothilal et al. [2020b] or the l_1 or l_2 distance metric (MAD) proposed by Wachter et al. [2018].

4.4 Summary & Conclusion

To compare and contrast the different approaches described in this chapter, the Table 4.1 gives the advantages and disadvantages of each approach. By conducting a comparative analysis of the strengths and weaknesses of the XAI approaches, the following conclusions can be made,

- Among the feature contribution methods, SHAP-based explainers such as KernelSHAP and TreeSHAP are completely model-agnostic, supporting a wide range of ML techniques (Linear and non-linear models).

- Among the SHAP-based explainers, TreeSHAP has very high computing efficiency for tree-based models. As mentioned in the previous chapter, tree-based ensemble models have high performance and are widely used for classification tasks. Unlike KernelSHAP, TreeSHAP can provide global and local explanations by taking into account individual and combined feature effects. Since TreeSHAP provides faster SHAP estimation than kernel shape, TreeSHAP has been chosen for further investigation for the proofs of concepts demonstration.
- Integrated Gradients (IG) only supports ANN models and does not provide global feature importance. Since the objective is to investigate both global and local explanations, IG won't be considered for further investigation.
- Among the example-based methods, Diverse Counterfactuals are the state-of-the-art approach for generating counterfactual local explanations. Counterfactuals can be generated taking user context into considerations. It is also more interesting to investigate diverse counterfactuals as it is under-researched and no research has been found in the literature for improving the interpretability of ML/ANN-based fraud detection models using Diverse Counterfactuals.

Explainability Approach	Algorithm	Supported Models	Advantages	Disadvantages
Feature Contribution	Shapley Value	Model Agnostic	-Based on a solid theoretical foundation (Game Theory) -Provides contrastive explanations	-very low computational efficiency -Easy to misinterpret -No ability to provide explanations for a subset of features -Need access to the entire dataset for computing new Shapley values -Provides only local explainability
Feature Contribution	KernelSHAP	Model Agnostic	-Based on a solid theoretical foundation (Game Theory) -Provides contrastive explanations -Provides global explainability	-Slower but has better computing efficiency than Shapley value approach -Does not take into account feature interaction effects. -Need access to the entire dataset for computing new SHAP values
Feature Contribution	TreeSHAP	Model Agnostic among Tree-based models	-Based on a solid theoretical foundation (Game Theory) -Provides global explainability -Provides contrastive explanations -TreeSHAP has very high computing efficiency for tree-based models -Accounts for feature interactions effects -Do not need access to the full dataset for computing new SHAP values	-Only used for tree-based methods (So not completely model-agnostic) - Can produce unintuitive feature contribution scores - Only tabular data
Feature Contribution	Integrated Gradients	Model Agnostic among ANN models	- Can handle image, text and tabular data - Easy to implement - Very simple theoretical foundations - High computational efficiency	- Does not provide global feature importance - Does not take into account feature interaction effects like TreeSHAP
Example-based method	Classical Counterfactuals	Model Agnostic	- Very simple explanations to model behavior - Easy to implement - Works for rule-based models as well	- Does not take into account user-context -Does not provide diverse & feasible counterfactuals - Does not provide feature sparsity in counterfactuals
Example-based method	Counterfactuals guided by prototypes	Model Agnostic	- Faster than classical counterfactuals - Provides diverse counterfactuals	- Harder to implement - Does not provide feasible counterfactuals
Example-based method	Diverse Counterfactuals	Model Agnostic	- Provides diverse and feasible counterfactuals - Provides feature sparsity - Takes into account user-context	- low computational efficiency - Currently works only for gradient-based differentiable models like XGBoost and MLP

Table 4.1: Summary of Explainable AI Approaches

5 Data

To obtain optimally performing Machine Learning (ML) and Artificial Neural Network (ANN) based fraud detection models, transaction datasets with optimal quality and granularity are required. There are various legal, regulatory, and security complexities with handling real-world financial transaction data from the financial institution sponsoring this research. Therefore, for the purpose of this thesis research, synthetically generated transaction data is used for training and evaluating the fraud detection models. Lopez-Rojas et al. [2016] developed an agent-based data simulator called PaySim, that generates synthetic data, mimicking normal and fraudulent transaction behavior. The PaySim simulator generates transaction datasets that are representative of real-world mobile-payments transaction data. The synthetic dataset generated using PaySim has several data quality (DQ) issues and this prompted an elaborate data pre-processing step involving Exploratory Data Analysis (EDA) and feature engineering.

In this chapter, Section 5.1 discusses the advantages and disadvantages of using synthetic data for fraud detection and AML research, Section 5.2 elaborates the data preparation process involving EDA and feature engineering, and Section 5.3 explains the modus operandi of fraud embedded in the synthetic transaction dataset.

5.1 Synthetic Data

The Dutch financial institution sponsoring this research generates large quantities of real-world transaction data from the millions of financial transactions happening within its infrastructure each year. Due to the stringent regulations imposed by the European Union through General Data Protection Regulation (GDPR), financial institutions have stringent policies around data protection, data use, and data sharing. Since GDPR imposes heavy fines in the order of tens of millions of euros if personal data found in transaction data is compromised. The financial institution has a legal and regulatory obligation to safeguard and protect customer transaction data and associated personal data such as customer name, address, and unique identifiers like credit card information found in it. Therefore, access to sensitive transaction data is hard to obtain, and access is given only to select individuals who are involved in business-critical projects. This thesis research does not qualify to be a business-critical project and therefore, access to real-world transaction data has not been provided by the financial institution.

When real-world datasets are hard to obtain, synthetic datasets can be used for fraud detection and AML research Lopez-Rojas and Axelsson [2012], Lopez-Rojas et al. [2016], Weber et al. [2018]. As a workaround for the lack of access to real-world transaction data from the financial institution, a synthetic transaction dataset generated using PaySim has been used for this thesis research. In the ML context, synthetic data is any labeled or unlabelled information that is generated using computer-aided simulations to provide an alternative to real-world data. Synthetic data can be anything from augmented data variations during model training to completely artificially generated datasets Wrenninge and Unger [2018]. PaySim is an example of the latter that generates entirely artificial transaction datasets using agent-based modeling and simulations.

5.1.1 Advantages of Synthetic Data

The main advantage of using synthetic data is that various payment fraud and money laundering scenarios can be simulated under user-defined contexts. Simulation-based data generators like PaySim can

simulate endless possibilities of fraudulent and money laundering transactions Lopez-Rojas and Axelsson [2012]. Therefore ML models can be trained to identify novel fraudulent and money laundering patterns that are not available in historical real-world transaction data. It also enables researchers to compare and contrast different fraud detection models trained using the same dataset. This can help researchers identity a benchmark model for performance testing of real-world transaction monitoring systems Barse et al. [2003]. Therefore, synthetic data also enables the evaluation and validation of models trained using historical transaction data. Tremblay et al. [2018] suggests that deep learning models trained using artificially generated data can perform better than models trained using real-world data. Therefore, synthetic datasets enable researchers in the fraud detection domain to develop highly performing models that can significantly reduce the False Negative Rate (FNR) without the need for large quantities of real-world transaction data. According to Barse et al. [2003], a significant advantage of synthetic data generators is that it enables rapid experimentation of fraud detection models under different system parameters and conditions. Synthetic data generators like PaySim enable the simulation of the following data properties that are significant for training and testing fraud detection models Lopez-Rojas et al. [2016] Barse et al. [2003],

- Labelled data is need for supervised ML tasks.
- The ability to embed different fraudulent transaction scenarios in the synthetic data
- The ability to control data skew (fraudulent/genuine transaction ratio) in the data distribution of the model.
- The ability to generate large volumes of transaction data as training, evaluating, and validating ML models is data intensive.
- The statistical properties of the simulated data should be similar to real-world datasets. PaySim can only simulate mobile-payments transactions between two accounts within the same bank.

5.1.2 Disadvantages of Synthetic Data

The multitude of fraudulent transaction scenarios occurring in the real world cannot be realistically simulated and subsequently generated using a single type of simulation-based synthetic data generator Lopez-Rojas et al. [2016]. The feature space of the simulated synthetic datasets is limited to the simulation capabilities of the data generator. Therefore, feature space in synthetic datasets is smaller in dimension and they are usually not close approximations of real-world transaction datasets Lopez-Rojas et al. [2016]. It will take considerable development effort and time to alter the simulation model to generate data with feature space that is a close approximation to the real-world datasets. The quality of the dataset is heavily impacted by the quality of the simulation model. For example, PaySim lacks the ability to control data skew or class distribution ratios in the dataset. The inability to control the data skew is an important data-quality issue considering the real-world transaction datasets have widely varying data skew ratios Barse et al. [2003]. Finally, user acceptance of synthetic datasets in training ML models for use in business-critical applications can be limited Bellovin et al. [2019].

5.2 Data Preparation

Generating synthetic transaction data using the PaySim simulator [1] is fraught with failure as the developers don't provide proper support for utilizing the simulator. Fortunately, the developers of PaySim have uploaded a transaction dataset generated using the simulator to the popular code repository - kaggle [2].In this section, the methodology for analyzing and improving the data quality issues in the synthetically generated PaySim transaction dataset obtained from Kaggle has been described.

[1] https://github.com/EdgarLopezPhD/PaySim
[2] https://www.kaggle.com/ealaxi/paysim1

5.2.1 Exploratory Data Analysis

The PaySim dataset has 11 features with 6,362,620 transactions. There is a very high imbalance between the majority (genuine transactions) and minority (fraudulent transactions) classes. The class imbalance is denoted by a data skew of 0.0013% for this dataset. Among the 6,356,620 total transactions, 8213 transactions are fraudulent. The very high imbalance is problematic as it can affect the classification performance of the trained ML models. There are several ways to tackle the class-imbalanced datasets, starting with reducing the feature space and number of observations. In Table 5.1, a detailed description of the features in the PaySim transaction dataset is given. There are five main transaction types in the dataset and they are described as,

1. **Debit**: money is moved directly into a bank account through this transaction.
2. **Transfer:** money is sent to another bank account from a client (C).
3. **Cash-in:** money moves through the transaction network through a merchant (M).
4. **Cash-out:** money moves out of the transaction network through a merchant (M).
5. **Payment:** money exchanges for goods or fungible entities in this transaction type.

Feature Space	
Features	Feature Description
step	one step is equivalent to one hour in real-world.
type	Transaction type
amount	Amount transacted.
nameOrig	Originator account of the transaction.
oldbalanceOrig	Existing balance in the originator account before completing the transaction.
newbalanceOrig	Existing balance in the originator account after completing the transaction.
nameDest	Destination account of the transaction
oldbalanceDest	Existing balance in the destination account before completing the transaction.
newbalanceDest	Existing balance in the destination account after completing the transaction.
isFraud	Fraud committed by a malicious actor to move funds from one account to another.
isFlaggedFraud	A rule-based system that flags perceived fraudulent transaction, where transaction amount $\geq$200.000 in a given step.

Table 5.1: Description of features in the PaySim dataset as publicly mentioned in "Synthetic Financial Datasets for Fraud Detection" dataset in Kaggle.com

Figure 5.1 gives the number of transaction records per transaction type. The type "debit" has the lowest number of transactions while "cash-out" has the highest. Most importantly, fraudulent transactions only exist in "cash-out" and "transfer" types. There are zero fraudulent transactions in other transaction types.

Table 5.2 gives the count of genuine and fraudulent transactions in the transaction dataset. The transaction records belonging to 'CASH IN'. 'DEBIT', 'PAYMENT' are dropped from the dataset as they don't have any fraudulent transactions belonging to them. These transaction types are dropped from the dataset as they won't provide any relevant information on the pattern of fraudulent transactions during the ML model training process. Moreover, dropping these transaction types has improved the 'data skew' or 'class imbalance from 0.0013% to 0.003%.

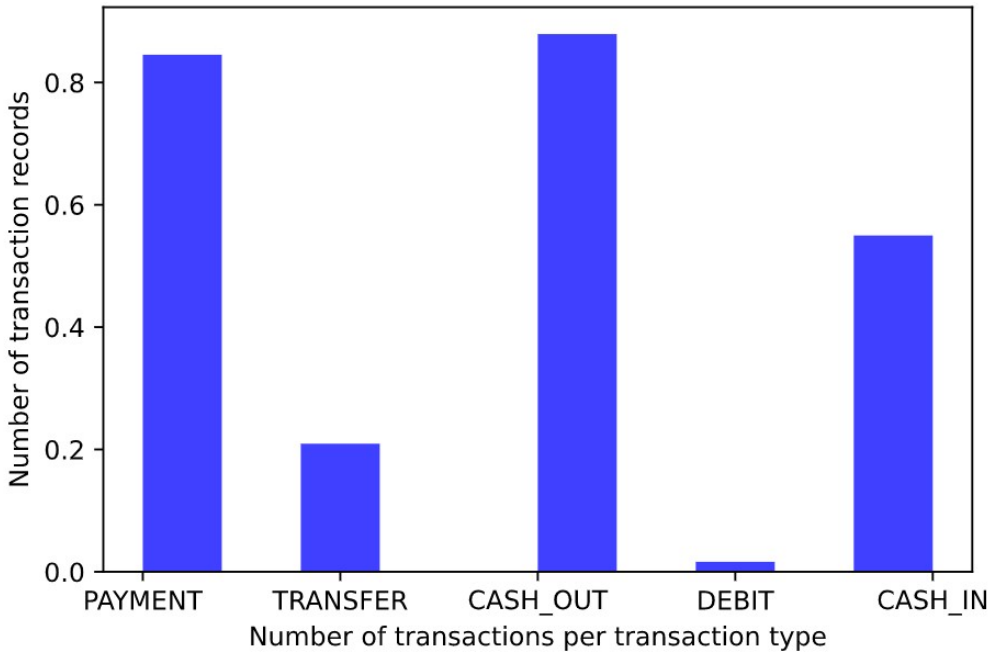


Figure 5.1: Number of transactions per transaction type in the PaySim dataset

Transaction Type	Genuine Transactions	Fraudulent Transactions	Total
CASH IN	1399284	0	1399284
CASH OUT	2233384	4116	2237500
TRANSFER	528812	4097	532909
DEBIT	41432	0	41432
PAYMENT	2151494	0	2151494
TOTAL	6354407	8213	6362620

Table 5.2: Distribution of genuine and fraudulent transactions in PaySim dataset

To further improve the class imbalance in the dataset,several irrelevant features are removed due to the following reasons,

- **nameOrig & nameDest** - According to the description of the PaySim simulator as shown in Table 5.1, 'nameOrig' & 'nameDest' features denotes the customer or merchant ID involved in the transaction. Customer ID is prefixed with a 'C' and merchant ID is prefixed with an 'M'. Even though the description mentions that there is a mix of merchant and customer IDs in the dataset, the actual generated dataset has only customer IDs with the prefix 'C'. In the generated dataset, the two features only point to transactions involving customers and do not differentiate between merchants and customers. Therefore, the nameOrig and nameDest features are dropped from the feature space during the data pre-processing step.
- **isFlaggedFraud** - According to the description of the PaySim simulator as shown in Table 5.1, the "isFlaggedFraud" feature is true when the transaction amount is greater than EUR 200,000. When the amount is greater than EUR 200,000, the transaction is immediately flagged as fraudulent and stopped. In the actual dataset, this scenario is true only for 16 transactions among more than 6 million transactions. It is important to note that these 16 'flagged' transactions are also set to be true in the 'isFraud' feature. The 'isFlaggedFraud' feature has to be very insignificant when it comes to predicting fraud and therefore it is dropped from the feature space during the data pre-processing step.

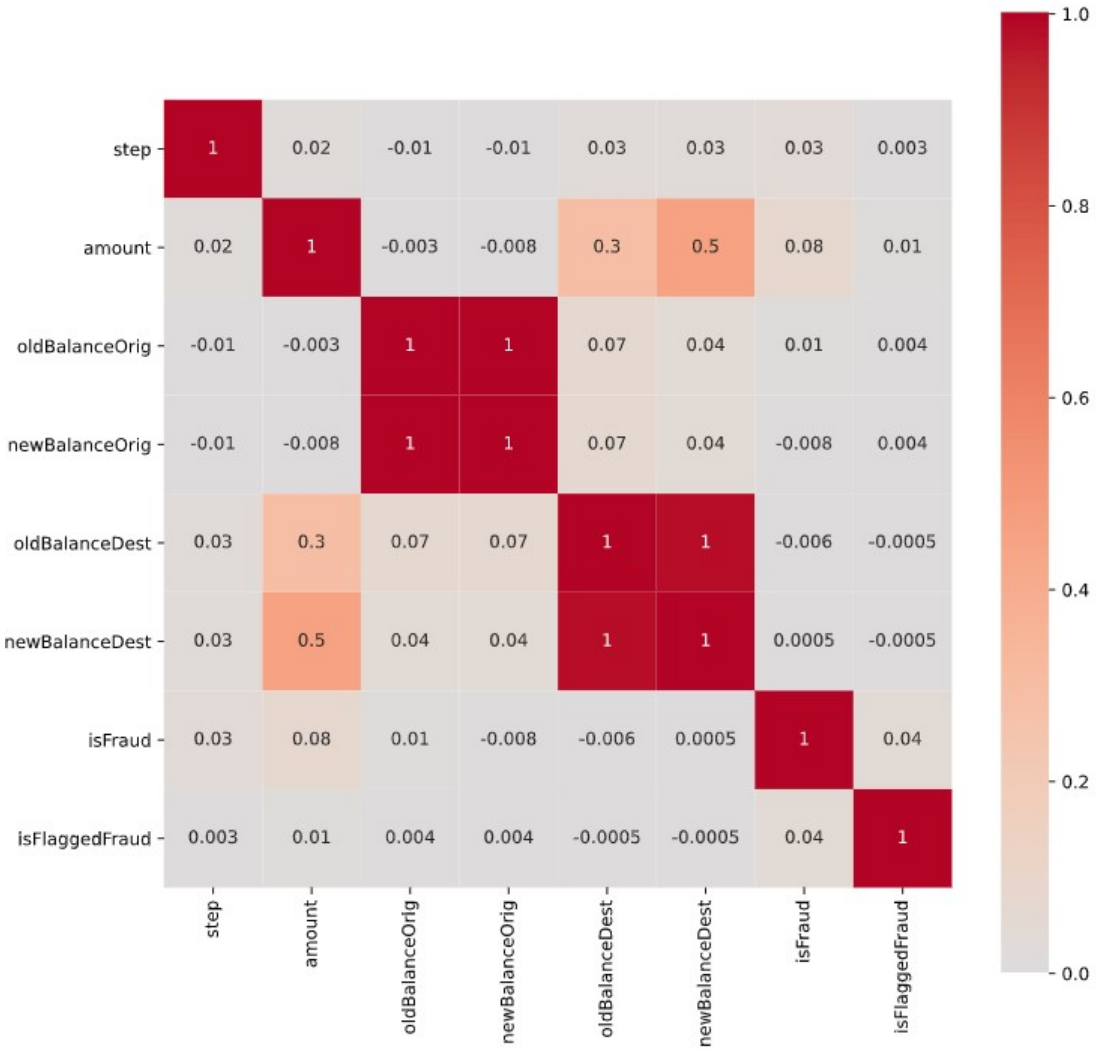


Figure 5.2: Correlation heat-map of PaySim feature space

There are a couple of interesting observations from the Pearson co-efficient heat-map in Figure 5.2,

- The transaction 'amount' feature is weakly correlated with the 'isFraud' feature as the correlation coefficient is only 0.08. This means that the 'amount' feature cannot be the only good predictor of fraud but it could contribute to the prediction of fraud. Dropping 'isFlaggedFraud' seems to be a good decision as it is set to true only based on the 'amount' feature.
- The 'amount' feature has a moderate correlation of 0.3 and 0.5 with 'oldbalanceDest' and 'newbalanceDest' respectively. For this reason, the amount feature has not been dropped along with 'newbalanceDest' and 'oldbalanceDest'

5.2.2 Feature Engineering

The generated dataset has several data quality issues with the following features, 'amount',

1. 'oldbalanceOrig'
2. 'newbalanceOrig'

3. 'oldbalanceDest'
4. 'newbalanceDest'

The data quality issues concerns about 50% and 1.2% of fraudulent and genuine transactions respectively where the transaction amount is non-zero while the 'oldbalanceOrig','newbalanceOrig','oldBalanceDest' and 'newBalanceDest' features are zero. The discrepancy in these amount balance features can severely impact the performance of the trained ML models. Therefore, imputation of missing values is done along with the creation of two new error features to account for the amount balance errors in the dataset. The two error features are created using the Equation 5.1 & Equation 5.2 as shown below,

$$errorBalanceOrig = newBalanceOrig + amount - oldBalanceOrig \tag{5.1}$$

$$errorBalanceDest = oldBalanceDest + amount - newBalanceDest \tag{5.2}$$

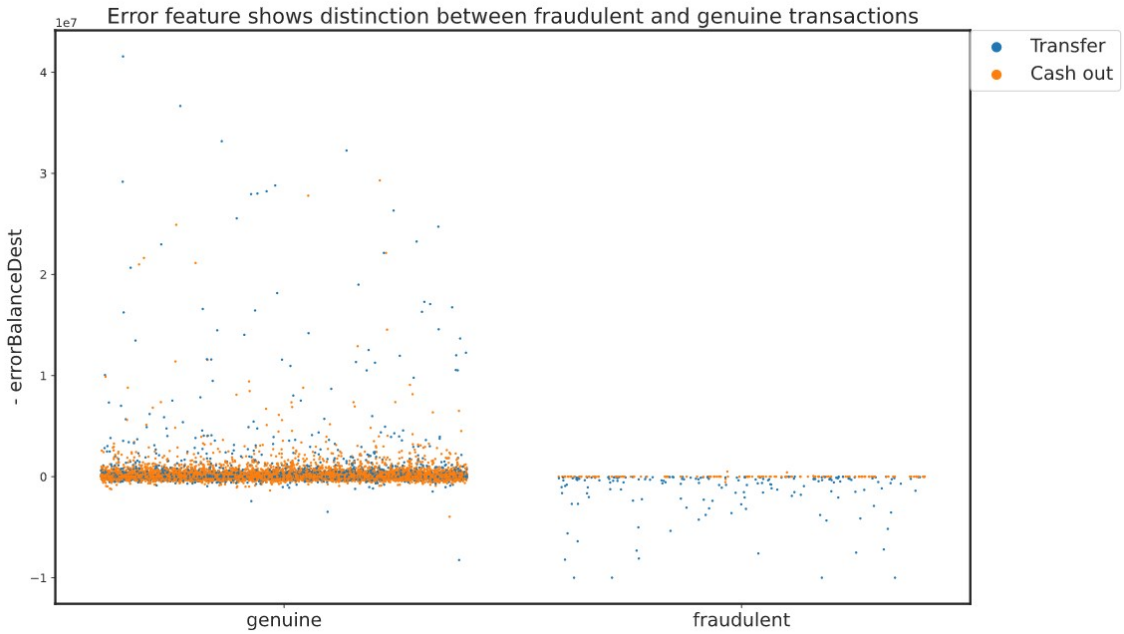


Figure 5.3: Plot shows error feature separating fraudulent transactions from genuine transactions

It is remarkable to see in Figure 5.3, that the 'errorbalanceDest' feature classifies fraudulent transactions distinctly from genuine transactions. The 'oldBalanceDest' and 'newBalanceDest' features with zero values are considered as missing values and "NaN" is imputed for these values.

One-hot encoding is done to change the categorical feature 'type' from 'string' type to 'int32' type (Binary). The type "Transfer" is zero while "Cash-out" is one. To make sure that the features have similar feature weights while training the model, a numerical standard scaling technique is done.

5.3 Modus Operandi of Fraud

According to Lopez-Rojas et al. [2016], the modus operandi of fraud that is synthetically embedded in the PaySim transaction dataset is shown in Figure 6.1. A transaction of type "Transfer" is made to a criminal account 'B' from genuine account 'A'. The holder of the fraudulent account 'B' is the alleged

Figure 5.4: Modus Operani of Fraud in PaySim dataset

criminal committing the transaction fraud. Once the money is accrued in the criminal account 'B' then it is cashed out to a third-party account 'C' through a transaction of type 'Cash Out'. This implies that the criminal account is the originator of a 'Cash Out' transaction and the destination of a 'Transfer' transaction. It has to be noted that all three accounts belong to the same banking entity but account 'C' could also belong to a merchant such as PayPal or Amazon. However, the modus operandi of fraud in the actual dataset could not be confirmed to have the pattern of 'Transfer' being followed by 'Cash Out'. There are no transactions where the 'nameDest' (destination) of a 'Transfer' labelled as fraudulent is the 'nameOrig' (Originator) of 'Cash Out' labelled as fraudulent. Based on EDA, there are three accounts (C1175896731,C2140495649,C2029041842) where the 'nameDest' (destination) of a 'Transfer' labelled as fraudulent is the 'nameOrig' (Originator) of 'Cash Out' labelled as genuine. Therefore, these three 'Cash Out' transactions could have been wrongly labelled as genuine or it simply means that the modus operandi of fraud is not explicitly reflected in the dataset. Further investigations of the trained ML models using Explainable AI approaches can shed more light on the nature of the modus operandi of fraud in the transaction dataset.

6 Detecting Transaction Fraud using Machine Learning & Artificial Neural Network (ANN) Models - Model Training and Evaluation

As discussed in Section 3.1.4, Machine Learning (ML) and Artificial Neural Network (ANN) models provide better performance in detecting financial transaction fraud than rule-based systems. However, due to the black-box nature of these models, it becomes sensitive to use them for high-risk banking use-cases. As discussed in Section 3.1.5, using ML and ANN models in high-risk banking use-cases has a lot of regulatory and legal implications. Therefore, the financial institution sponsoring this research wants to improve the explainability and interpretability of fraud detection models by investigating Explainable AI approaches. To achieve this objective, two research phases have been proposed. Firstly, ML and ANN models have been trained and evaluated for detecting fraud using the PaySim financial transaction dataset (Refer Chapter 5). The first phase sought to develop optimally performing ML and ANN models that are trained and tested under various experimentation scenarios. In the second phase, Explainable AI approaches proposed in Chapter 4 are investigated using the fraud detection models developed in the first phase of the research. The first and second phases of the research are discussed in Chapter 6 and Chapter 8 respectively while the model results are discussed in Chapter 7. To accomplish the first phase of the research, the following research objective has been formulated,

Investigate the extent to which ML and ANN-based fraud detection models trained using synthetic transaction data detect financial transaction fraud

Figure 6.1: Two research phases involved in achieving the main research objective discussed in Section 3.1.6

In this chapter, Section 6.1 and Section 6.2 covers the methodologies for training and evaluating ML and ANN-based fraud detection models respectively.

6.1 Detecting Fraud using Machine Learning Models - Methodology

6.1.1 Motivation

The results of EDA in Chapter 5, show that the PaySim transaction dataset has a high class-imbalance between the genuine and fraudulent transactions. ML and ANN models trained using high class-imbalanced datasets are prone to poor performance as they don't sufficiently capture the background data distribution of the under-represented fraudulent class. Therefore, in addition to training the ML models for the fraud detection task, techniques like resampling and class-weighting have to be performed to reduce the effect of high class-imbalance on the performance of the trained models. Recent

research by Manju et al. [2019] shows that tree-based ensembles such as XGBoost and LightGBM (Refer Section 2.1.3) are better for binary classification tasks like fraud detection that involves high class-imbalanced datasets. In addition to handling class-imbalanced datasets, these boosting tree methods have faster computation speeds with reduced memory usage. The XGBoost models have the following advantages over other tree-based ensembles like LightGBM,

- XGBoost is computationally efficient as it can be distributed and trained in parallel Chen and Guestrin [2016].
- XGBoost is scalable so it can handle increased data-load Chen and Guestrin [2016].

Moreover, the boosting tree algorithm like XGBoost is supposed to have high classification performance comparable to ANN models Ponomareva et al. [2017]. Therefore, given the requirement of high classification performance for fraud detection tasks, XGBoost and LightGBM models are to be investigated for the fraud detection task. Random Forest (RF) classifiers are simpler versions of tree-based ensembles like XGBoost. Although Random Forest classifiers provide less performance than XGBoost models, RF is easy to interpret due to the simpler architecture. The inner working of the Random Forest classifier can be broken down into simpler decision tree rules to facilitate better explainability and interpretability. Random Forest performs well with datasets involving a large feature space. Given that real-world fraud detection systems, need better interpretability and the ability to handle large feature space, Random forest classifiers are to be investigated.

According to Oza [2018], Principle Component Analysis (PCA) reveals that the PaySim transaction dataset for the "Transfer" transaction type is linearly separable between genuine and fraudulent transactions. Therefore, ML algorithms that can linearly separate predictions such as Logistic Regression and linear Support Vector Machine (SVM) classifiers are to be investigated. To evaluate the models, a baseline scikit-learn stratified dummy classifier is trained. The simple baseline model is used as a benchmark to evaluate the performance of the XGBoost, lightGBM, Random Forest, and Logistic Regression classifiers. In the Chapter 5, EDA along with feature engineering is done to mitigate the class imbalance in the transaction dataset. In this section, the class imbalance is tackled in the process of model training using resampling and class weighting techniques. Hyperparameter optimization involving state-of-art techniques such as Bayesian Optimisation has been done to improve model performance. Different experimentation scenarios, involving a combination of hyperparameter optimization, resampling, and class-weighting are done to understand the most suitable approach for increasing model performance in fraud detection tasks. In Section 6.1.5, the model evaluation metrics that are used to evaluate the performance of trained ML fraud detection models are discussed.

In conclusion, six ML models (*Dummy Classifier, Logistic Regression, Linear SVM,Random Forest,XGBoost, LightGBM*) are trained using the preprocessed PaySim transaction dataset under different experimentation scenarios.

6.1.2 Dataset

The various data prepossessing and feature engineering steps elaborated in Chapter 5 have been completed to obtain a transaction dataset suitable for training six different ML models. The missing values in the transaction dataset are imputed and the dataset is scaled using standard scaling. The standard scale value for any sample is by Equation 6.1,

$$z = \frac{(x - \mu)}{s} \tag{6.1}$$

The pattern of fraud in one transaction type (Cash Out) could be influenced by the pattern of fraud in another transaction type (Transfer) as mentioned in the 'Modus Operandi of Fraud' in Section 5.3. Therefore two subsets of the entire preprocessed transaction dataset are created as shown in Table 6.1. ML models trained using the two subsets of transaction dataset can reveal the pattern of fraud embedded in the data.

Dataset Type	Genuine Transactions	Fraudulent Transactions	Total	Data Skew (%)
Full Dataset	2762196	8213	2770409	0.003
Transfer Dataset	528812	4097	532909	0.008
Cash Out Dataset	2233384	4116	2237500	0.002

Table 6.1: Subsets of preprocessed transaction dataset

6.1.3 Hardware & Software Environments

Python 3.7 is used for EDA, data pre-processing, feature engineering, model training, and evaluation. Several Python libraries such as Pandas 1.2.4, Matplotlib 3.4.1, Seaborn 0.11.1, Numpy 1.20.2 are used for EDA, data pre-processing, and, feature engineering. Scikit-learn 0.24 Python library is used to train the ML models except for the XGBoost model. The XGBoost model has been trained using Python library XGBoost 1.4.1 and hyperparameter optimization or tuning has been done using python packages Hyperopt and Ray 2.0.0. The model training and evaluation are performed in an HP workstation using an Intel(R) Core(TM) i7-6700HQ CPU clocked at 2.60GHz with 16 GB of RAM.

6.1.4 Machine Learning Model Training & Hyperparameter Optimisation

The Full, Transfer, and Cash Out datasets are split into training data with 80% of random observations and test data with 20% of random observations. The 80/20 split seems to have a better representation of both positive and negative classes in training and testing datasets than a 70/30 split. This has been concluded from experimentation using both versions of train/test split. The ML models is trained using the training data to detect fraud while the testing dataset is used to evaluate the trained ML models for their classification performance. To optimally solve the machine learning tasks at hand, hyperparameter optimisation or tuning is done. The hyperparameter for ML algorithms is a parameter that controls the training process. On the otherhand, model parameters are inherent configurations of the ML algorithm that is learnt during the training process. Model parameters are configured by fitting a model to the data while hyperparameters are learnt by the tuning process. The tuning process is an optimisation approach where-in a loss function is minimized to learn the optimal set of hyperparameters. Kong et al. [2019] concludes that compared to default hyperparameters, a combination of hyperparameter tuning and resampling approaches produce optimal performance in ML models trained using class imbalanced datasets.

To investigate how hyperparameter optimisation can help with improving performance in ML models for the fraud detection task, various experimentation scenarios are formulated in Table 6.2.

Scenario	ML models	Hyperparameter Optimisation	Resampling Approaches
Scenario 1	Dummy Classifier & Logistic Regression	Default Hyperparameter	No
Scenario 2	Linear SVM, Random Forest & LightGBM	Optimised Hyperparameter	No
Scenario 3	XGBoost	Optimised Hyperparameter	Yes

Table 6.2: Different Scenarios of ML Experimentation

Three scenarios of ML experimentation for the fraud detection task have been formulated. In scenario 1, ML models are trained using default hyperparameters with no application of resampling approaches. In scenarios 2 & 3, ML models are trained with optimal hyperparameter with and without applying resampling approaches. The hyperparameters for Linear SVM, Random Forest, and LightGBM are tuned using GridSearchCV and for XGBoost using Bayesian Optimization (BO) algorithms. According to Ranjan et al. [2019] Grid Search Cross-Validation (GridSearchCV) is a hyperparameter optimisation algorithm that is used to train and evaluate an ML model for every combination of model parameters specified in a grid search space. The output of the optimisation process over the grid search space is the optimal

hyperparameters for the model being trained for a specific task. The hyperparameters for the XGBoost model are tuned using Sequential model-based optimisation (SMBO) or Bayesian Optimisation (BO). According to Putatunda and Rama [2018], Bayesian Optimisation is a highly efficient model prediction function optimisation technique for computationally intense ML models such as XGBoost.Two different python libraries - Ray & Hyperopt, are used to find the optimal hyperparameters using Bayesian optimisation. The different models trained under the three different scenarios (Refer Table 6.2) are evaluated using the same set of evaluation metrics described in Section 6.1.5. By having a common set of classification evaluation metrics, the performance of the different combinations of hyperparameter optimisation and resampling approaches on models trained using class imbalanced datasets can be investigated. Based on the credit card fraud detection case-study by Meng et al. [2020], a resampling approach named SMOTE (Synthetic Over-Sampling Technique) improved the fraud detection performance of XGBoost. Therefore, SMOTE is the chosen resampling approach for XGBoost.

The following hyperparameters (Default & Optimal) are used for training the shallow ML models,

- **Dummy Classifier:** A dummy classifier is a simple, baseline classifier model that is used as a benchmark to study the performance of advanced classifier models. If an advanced model is performing lesser than the bench mark model, then the advanced model has to be optimised for the fraud prediction task using hyperparameter optimisation. The baseline model is a sci-kit learn stratified dummy classifier where the hyperparameter ('strategy') is set to 'stratified'.

- **Logistic Regression:** The hyperparameter ('regularization' (C)) is set to the default value ($C = 1$) as the PaySim transaction dataset is linearly separable as concluded by Oza [2018] . Refer Section 2.1.3 for a detailed description of Logistic Regression.

- **Linear SVM:** The hyperparameter ('regularization' (C)) is set to the default value ($C = 1$) as the PaySim transaction dataset is linearly separable as concluded by Oza [2018]. Refer Section 2.1.3 for a detailed description of Linear SVM.

- **Random Forest:** The 'max_depth' is set to 3 with 'n_jobs' set to 3.The 'max_depth' parameters denote the maximum number of splits in the training process while 'n_jobs' denotes the number of computational jobs running in parallel. Refer Section 2.1.3 for a detailed description of Random Forest.

- **XGBoost:** The 'scale_pos_weight' parameter is set to the total number of genuine divided by the fraudulent transactions as shown in Equation 6.2. A ratio greater than one can make the algorithm converge faster to a solution when training using a high class imbalanced dataset. Refer Section 2.1.3 for a detailed description of XGBoost.

$$Class\ Weight = \frac{\sum genuine\ transactions}{\sum fraudulent\ transactions} \tag{6.2}$$

Table 6.3 gives the description of hyperparameters of the XGBoost-based fraud detection model. These hyperparameters have been tuned using a hyperparameter optimisation algorithm called Bayesian Optimisation. Bayesian Optimisation algorithm (BO) is a "sequential hypothesis testing"-based strategy that optimises for better model solution by taking into account prior information of the model Pelikan et al. [1999].

Hyperparameters of XGBoost	Description
max_depth	This hyperparameter denotes the "maximum tree depth" of the decision trees internally used by XGBoost to aggregate predictions of an instance. It affects model complexity as shorter trees can underfit while larger trees can overfit. The default value is 3.
min_child_weight	At each new leaf node of the decision tree, the sample available at the node is split into two groups based on implicit decision rules. It is possible to constraint the split if there are only a few samples are available at the node. It impacts model complexity and the default value is 1.
colsample_bytree	When constructing a new decision tree, the subsample size of the features in the training dataset is used.
gamma	This hyperparameter denotes the minimum loss reduction needed for a split of the samples in the leaf node. It controls the model complexity.
reg_alpha	Refers to the L1 regularization term. It mitigates overfitting of the model.
reg_lambda	Refers to the L2 regularization term. It is used to reduce overfitting of the model.
n_estimators	Refers to the number of "boosting rounds". It determines the number of decision trees to be trained. The default value is 10.

Table 6.3: Description of the tuned hyperparameters in the XGBoost model.

Table 6.4 gives the optimal hyperparameters for the XGBoost model obtained using the Bayesian Optimisation process.

Datasets	colsample_bytree	gamma	max_depth	min_child_weight	reg_alpha	reg_lambda	n_estimators
Full dataset	0.641	8.212	5	1	130	0.844	180
Transfer	0.892	5.979	7	0	56	0.349	180
Cash Out	0.844	8.77	16	3	42	0.496	180

Table 6.4: Hyperparameters for XGBoost

- **LightGBM**: The 'num_leaves' parameter is set to 30 and the 'objective' parameter is set to 'binary'. Refer Section 2.1.3 for a detailed description of LightGBM.

6.1.5 Machine Learning Model Evaluation Metrics

The performance of the shallow machine learning models under the three scenarios have been evaluated using the metrics below and the results are summarised in Chapter 7 under Section 7.1.

- **Accuracy:** It gives the ratio of correctly classified outcomes to the total number of outcomes classified by the model. Due to the class imbalance in the dataset, accuracy metrics can be misleading. The minority class is a significantly low percentage of the total data, any wrong classification in the minority class will have very little contribution to the total accuracy score. Even though it can be misleading, the accuracy metric is suitable for contrasting the performance with the benchmark model. Equation 6.3 is the formula for calculating accuracy where TP, TN,FP & FN refers to "True Positives", "True Negatives", "False Positives" and "False Negatives" respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.3}$$

- **Confusion Matrix:** It gives the distribution of the transactions between TP, TN, FP, FN in a matrix format.
- **Precision**: It is the ratio of true positives (TP) to the sum of true positives (TP) and false positives (FP) Davis and Goadrich [2006] (Refer Equation 6.4). It denotes the capability of the classifier to correctly identify fraudulent transactions. A high precision value denotes a low false-positive rate(FPR). Precision answers the following question - Of all transactions that are labelled as fraudulent, how many are actually fraudulent?

$$Precision = \frac{TP}{FP + TP} \tag{6.4}$$

- **Recall:** It is the ratio of true positives (TP) to the sum of true positives (TP) and false negatives (FN) Davis and Goadrich [2006] (Refer Equation 6.5). A high recall rate denotes a low false-negative rate (FNR). Recall answers the following question - Of all the transactions that are actually fraudulent, how many are labelled as fraudulent? Recall is also called as 'Sensitivity'.

$$Recall = \frac{TP}{FN + TP} \tag{6.5}$$

- **F1 score:** The F1 score combines precision and recall into one value by taking an harmonic mean (Refer Equation 6.6). F1 is a better predictor of performance that accuracy in models trained using class imbalanced datasets.

$$F1 = \frac{(Precision * Recall) * 2}{precision + recall} \tag{6.6}$$

- **Area Under the Precision-Recall Curve(AUPRC):** AUPRC denotes a trade-off between precision and recall. A low AUPRC denotes low precision and recall values. AUPRC captures the trade-off between high TPR and low FPR. AUPRC gives the performance of the model to correctly detect fraudulent transactions.
- **k-fold cross-validation (CV):** To evaluate the models for generalisability across new datasets, five fold cross-validation is done. This technique splits the entire dataset into five equal datasets. For each iteration of model training, One out of five datasets is used for testing while the others are used for training.

6.2 Detecting Fraud using ANN models - Methodology

6.2.1 Motivation

In the previous Section 6.1, several Machine Learning models such as Random Forests, XGBoost, LightGBM, Logistic Regression are used for fraud detection using the synthetically generated mobile-payments transaction data. In addition to these binary classification models, Artificial Neural Network-based Multi-Layer Perceptron (MLP) models have been proved to perform well in fraud detection and anti-money laundering (AML) tasks Mubalaike and Adali [2017]. MLP is a class of "feed-forward ANN models" (Mubalaike and Adali [2017] & Mishra and Dash [2014]) that has multiple layers of interconnected nodes (perceptrons). There are input, hidden, and output layers Mubalaike and Adali [2017], each with a predefined number of nodes. The nodes are activated by a non-linear function based on the features captured in the input layer. During the model training process, MLP learns to approximate the

decision boundary of the model prediction function through the process of backpropagation. Backpropagation denotes a suite of optimization algorithms that helps MLP learn to identify fraud by minimizing a non-linear loss function. According to Lv et al. [2008], Mubalaike and Adali [2017], Mishra and Dash [2014], MLP has higher performance in detecting fraud than many of the ML models like Linear SVM, Random Forests, and Decision Trees. MLP improves accuracy, average precision-recall score and reduces both False-Negative Rate (FNR) and False-Positive Rate (FPR). This section elaborates on various MLP models that have been trained to detect fraud using the synthetic transaction dataset.

6.2.2 Dataset & Implementation Details

The full dataset as mentioned in Table 6.2 is used for training the MLP models. Similar to the experimentation conducted in Section 6.1.4, various experimentation scenarios are formulated in Table 6.5. These scenarios are used to investigate the performance of MLP-based fraud detection models using hyperparameter optimisation, class weighting and oversampling.

Scenarios	MLP Models	Hyperparameter Optimisation	Resampling Approaches
1	Baseline MLP	No	No
2	MLP model with hyperparameter tuning	Yes	No
3	MLP model with hyperparameter tuning + Oversampling	Yes	Yes
4	MLP model with hyperparameter tuning + Class Weighting	Yes	No

Table 6.5: Experimentation of MLP-based fraud detection models under various scenarios

6.2.3 Hardware & Software Environments

Python 3.7 is used for EDA, data pre-processing, feature engineering, model training, and evaluation. Tensorflow 2 Keras API is used for training and evaluating the MLP models. The hyperparameter tuning of the MLP models are done using Keras Tuner and HParams TensorBoard Dashboard. The model training and evaluation is performed in an HP workstation using an Intel(R) Core(TM) i7-6700HQ CPU clocking at 2.60GHz with16 GB of RAM.

6.2.4 ANN Model Training & Hyperparameter Tuning

To train and evaluate the MLP models, the full transaction dataset is split into train,validation and test datasets. Training dataset is used to train a model (Fit a model to the data) while validation dataset is used to evaluate the model training process during the hyperparameter tuning process. The testing dataset is finally used to evaluate the trained MLP model using the classification evaluation metrics described in Section 6.1.5. The three datasets are scaled within the range [0,1] using the standard scaling technique mentioned in Section 6.1.2. The dimensions of the three datasets are shown in Table 6.6.

Datasets	**Samples of Data (Features & Labels)**	**Size of Feature Space**
Training	1773061	10
Validation	443266	10
Testing	554082	10

Table 6.6: Dimensions of the Training, Validation and Testing Datasets

The Tensorflow 2 (TF2) implementation of the Keras high-level API has been used to build and train the three different MLP models under four different scenarios as mentioned in Table 6.5. All three MLP models are Keras sequential models built using the sequential class of Keras API. After building the

three models, training dataset is used for the training process while the validation dataset is used to validate the training process. The model training history is stored at various model checkpoints and plotted to understand the performance of the training process. During the training process, the hyperparamters mentioned in Table 6.7 are tuned for scenarios 2 & 3 using the 'Hyperband' optimisation algorithm from Keras tuner. The Hyperband optimisation algorithm is a novel hyperparameter optimisation algorithm that performs better than Bayesian Optimisation Algorithm (BO) Li et al. [2018]. Hyperband performs better than BOA by cleverly managing computing resources among randomly sampled hyperparameter space by using early-stopping strategy.

Hyperparameters of MLP models	**Description**
Batch Size	Refers to "the number of samples" or number of feed forward-pass required before the model parameters are updated using backpropagation. It can be 20,64 or higher.
Epoch	Refers to the number of times the model is trained using the entire training dataset. It is usually between 0 and 100 but can be as high as 1000.
Learning rate	Refers to the amount of change to the model weights during the training process. It is also referred to as "step-size". It usually ranges between 0 and 1
Nodes in input dense layer	Refers to the number of input nodes in the first dense layer of the sequential model.
Activation function in input layer	Activation functions in ANN are used to induce non-linearity during the model training process.
Drop-out rate	Dropout is a regularization process used in ANN to reduce overfitting. Using dropout, multiple ANN architectures are trained in parallel to find the optimal configuration for a layer or combination of layers.
Activation function in output layer	Similar functionality to activation function in input layer.
Loss function in Backpropagation	Loss function is minimized during backpropagation using optimization algorithms like Stochastic Gradient Decent (SGD).

Table 6.7: Description of hyperparameters of the MLP models Mubalaike and Adali [2017] & Mishra and Dash [2014]

The following are the model build configurations and hyperparameters of the MLP-based fraud detection models under the four different scenarios mentioned in Table 6.5,

Baseline MLP

Figure 6.2 gives the architecture of baseline MLP and Table 6.8 gives the hyperparameters for training the baseline MLP model.

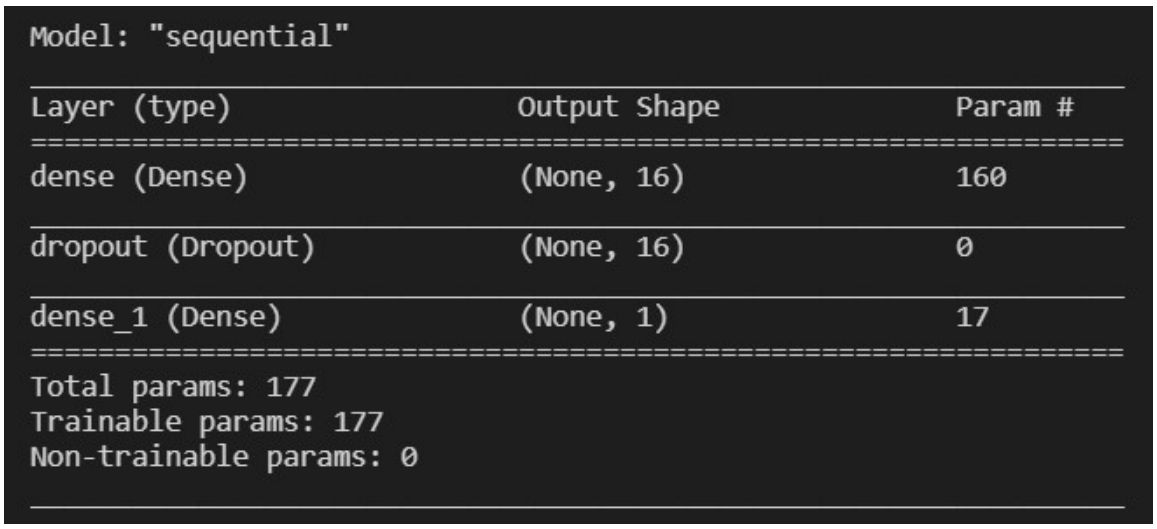


Figure 6.2: Architecture Summary of Baseline MLP

Hyperparameters	Value
Batch Size	500
Epoch	100
Learning rate	0.001
Nodes in input dense layer	16
Activation function in input layer	reLU
Drop-out rate	0.5
Activation function in output layer	sigmoid
Loss function in Backpropagation	Binary Cross Entropy

Table 6.8: Hyperparameters of Baseline MLP

MLP model with hyperparameter tuning & Oversampling: Figure 6.3 gives the architecture of scenario 2 MLP model and Table 6.9 gives the hyperparameters for training the scenario 2 MLP model. The hyperparameters have been learnt using the Hyperband optimisation process. For scenario 3, the same architecture and hyperparameters of the scenario 2 MLP model has been used. Scenario 3 is a combination of hyperparameter tuning and SMOTE oversampling. Synthetic Minority Over-Sampling Technique (SMOTE) is widely applied to improve the classification performance of ANN models that are trained using class-imbalanced datasets More [2016]. SMOTE oversamples the minority class (Fraudulent transactions) to improve the minority class representation.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 400)               4000
_________________________________________________________________
dropout (Dropout)            (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 401
=================================================================
Total params: 4,401
Trainable params: 4,401
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.3: Architecture summary of MLP models under scenario 2 & 3 as described in Table 6.5

Hyperparameters	**Value**
Batch Size	2048
Epoch	38
Learning rate	0.01
Nodes in input dense layer	400
Activation function in input layer	reLU
Drop-out rate	0.2
Activation function in output layer	sigmoid
Loss function in Backpropagation	Binary Cross Entropy

Table 6.9: Hyperparameters of MLP models under scenario 2 & 3 as described in Table 6.5

MLP model with hyperparameter tuning & Class weighting

The number of fraudulent transactions (positive class) in the dataset is very low compared to the genuine transactions (negative class). This class imbalance means that the MLP-based fraud detection model finds it very hard to find a model prediction function that is accurate and generalizable. Therefore, in-addition to SMOTE oversampling techniques, class weighting can be performed. Class weighting or importance weighting is done to make the MLP classifier model to heavily weight the positive class over the negative class, Importance weighting has shown to improve performance of ANN-based deep learning models Byrd and Lipton [2019]. Equation 6.7 & Equation 6.8 have been used to calculate the class weight for negative class (0.50) and positive class (168.66) respectively. The same hyperparameters used for scenario 2 & 3 have been used for training the MLP model in this scenario using the class weights.

$$weight_{for,0} = \frac{1}{\sum genuine\ transactions} \frac{total}{2} \tag{6.7}$$

$$weight_{for,1} = \frac{1}{\sum Fraudulent\ transactions} \frac{total}{2} \tag{6.8}$$

The evaluation of the model performance for the four different models have been performed using the following metrics - Accuracy, Precision, Recall, Confusion Matrix and AUPRC. The results have been discussed in the Chapter 7 under Section Section 7.2.

7 Model Results

In this chapter, evaluation metrics such as accuracy, AUPRC, precision, recall, F1 score, and five-fold cross-validation involving AUPRC have been used to measure "classification performance" of the fraud detection models developed in Chapter 6. The performance results of Machine Learning (ML) and Artificial Neural Network (ANN)-based fraud detection models trained using the full dataset (Refer Table 6.1) are shown. The full dataset contains the combined transactions of 'Transfer' and 'Cash Out' types.

Refer Section 6.1.5 for a detailed description of the model performance metrics used to evaluate the performance of ML and ANN models in this chapter.

7.1 Machine Learning Model Results

As shown in Table 7.1, both XGBoost and lightGBM classifiers have very high accuracy scores of 0.999. Since we are using class-imbalanced datasets for training the models, accuracy is not a good indicator of performance. AUPRC & F1 scores are better indicators of performance for the trained fraud detection models. AUPRC is slightly higher for the XGBoost model with optimal hyperparameters but the difference with LightGBM and XGBoost with default hyperparameters is insignificant. This means that all the three high-performing models (Highlighted in Table 7.1) are equally good at detecting fraud. As mentioned in Section 3.1.3, False Positive Rate (FPR) and False Negative Rate (FNR) don't have the same costs associated with them. Missing out on detecting actual fraud has severe financial and regulatory consequences for the financial institution compared to the cost of analyzing false positives. Given this context, the models are highly performing if they have sufficiently lower FNR even if the FPR is moderately higher.

The F1 score is perfect for LightGBM, as FP (False Positives) and FN (False Negatives) are zero and six respectively. Therefore LightGBM is by far the best performing model among all the shallow ML models. The F1 scores are similar for the XGBoost model with and without optimal hyperparameters. Precision and recall scores are perfect for LightGBM, meaning that the model is near perfect in detecting fraud. LightGBM has the lowest (zero) false positives of all other shallow ML models. XGBoost with and without optimal hyperparameters have the same number of FN. This means that both XGBoost and lightGBM classifiers are very good at avoiding genuine transactions flagged as fraudulent and avoid missing the detection of fraudulent transactions. These models if realized using real-world datasets can reduce the false positive and negative rates significantly compared to the rule-based models. Interestingly in Table 7.3, it can be understood from the cross-validation scores that XGBoost does better in terms of generalisability of the model predictions in comparison to the lightGBM classifier. XGBoost classifier has an AUPRC score of 0.99 for three out of five training iterations while the lightGBM classifier has an AUPRC of 0.99 only for one of the five training iterations.

Scenarios	Shallow ML Models	Accuracy	AUPRC	Confusion Matrix (TN;FP;FN;TP)
1	Baseline Classifier	0.981	0.003	550839;1573;1664; 6
1	Logistic Regression	0.997	0.212	551460;772;946;724
2	Linear SVM	0.995	0.161	550557;1855;820;850
2	Random Forest	0.998	0.626	552410;2;625;1045
2	**LightGBM**	**0.999**	**0.996**	**552412;0;6;1664**
1	**XGBoost (Default Hyperparameters)**	**0.999**	**0.998**	**552386;26;5;1665**
3	**XGBoost (Optimal Hyperparameters + Oversampling)**	**0.999**	**0.999**	**690468;34;5;2096**

Table 7.1: Model performance results of trained shallow ML models under three different experimentation scenarios. The scenario 3 model has been trained using a different train-test split ratio (0.25 instead of 0.2) and that explains the increase in total transactions. TN, FP, FN, TP refers to True Negatives, False Positives, False Negatives, True Positives respectively

Scenarios	Shallow ML Models	Precision;Recall;F1
1	Baseline Classifier	0.003;0.004;0.004
1	Logistic Regression	0.48;0.43;0.46
2	Linear SVM	0.31;0.51;0.39
2	Random Forest	0.998;0.634;0.773
2	**LightGBM**	**1;1;1**
1	**XGBoost (Default Hyperparameters)**	**0.984;1;0.990**
3	**XGBoost (Optimal Hyperparameters + Oversampling)**	**0.984;0.998;0.990**

Table 7.2: Model performance results of trained shallow ML models under three different experimentation scenarios

Shallow ML model	k-fold CV (Metric: AUPRC)
Stratified Dummy Classifier	0.00297417; 0.00296318; 0.00296335; 0.00296148; 0.00296152
Logistic Regression	0.42873295;0.44797467; 0.42819712; 0.42910799; 0.44283091
Linear SVM	0.60225234; 0.42383028; 0.60842618; 0.05033887; 0.81093398
Random Forest	0.93352813;0.90473806;0.92256768;0.91345361;0.93208267
XGBoost (Default Hyperparameters)	0.09776857; 0.99374099; 0.99506585; 0.3571594 ; 0.99723967
LightGBM Classifier	0.04449762; 0.01079685;0.05415675;0.28818101; 0.99698191

Table 7.3: k-fold Cross Validation Score for trained Shallow ML models. This metric has been used for only the baseline models as it is computationally expensive.

7.2 ANN Model Results

In this section, the same model performance metrics (except k-fold cross-validation) in the previous section are used for evaluating the performance of the ANN-based fraud detection models. The class of ANN used for this research is Multi-Layer Perceptron (MLP). As seen in Table 7.4 the scenario 2 MLP with optimal hyperparameters has the highest AUPRC of all scenarios. Since the cost for increased FN is higher for the fraud detection use-case, the MLP model in scenario 2 is not performing better than the baseline model. The FNR has slightly increased with a slightly reduced number of TP. This means that the scenario 2 model is under-fitting (Refer Figure 7.1) and the hyperparameter optimization process could be improved further. The most interesting observation is regarding both the scenario 3 and 4 MLP models with oversampling and class-weighting techniques respectively. Both oversampling and class-weighting try to increase the representation of fraudulent transactions in the dataset, so the models can

learn to better predict fraud. This seems to have worked, as the increased representations of the fraudulent class have decreased the number of FN substantially. The number of TP has improved substantially from scenarios 1 and 2. The models have got better at detecting truly fraudulent transactions. A massive disadvantage of the oversampling and class-weighting strategies is that the number of FP has increased substantially and most of the increase is at the expense of True Negatives (TN). Under the context of this use-case, scenarios 2 and 3 could be considered to have good performance but the sheer number of FP makes it unsuitable for real-world applications. Several improvements have to be made to both scenario 2 and 3 MLP models to decrease the false positive rate.

Scenarios	**ANN Models**	**Accuracy**	**AUPRC**	**Confusion Matrix (TN;FP;FN;TP)**
1	Baseline MLP (Default Hyperparameter)	0.999	0.916	552367;45;424;1246
2	MLP (Optimal Hyperparameter)	0.999	0.983	552406;21;439;1216
3	MLP (Optimal Hyperparameters + Oversampling)	0.980	0.891	541354;11058;14;1656
4	MLP (Optimal Hyperparameters + Class Weighting)	0.987	0.645	545165;7247;80;1590

Table 7.4: Model performance results of trained ANN models under four different experimentation scenarios

Scenarios	**ANN Models**	**Precision**	**Recall**	**F1 Score**
1	Baseline MLP	0.965	0.746	0.841
2	MLP (Optimal Hyperparameter)	0.983	0.734	0.840
3	MLP (Optimal Hyperparameters + Oversampling)	0.130	0.991	0.230
4	MLP (Optimal Hyperparameters + Class Weighting)	0.180	0.952	0.303

Table 7.5: Model performance results of trained ANN models under four different experimentation scenarios

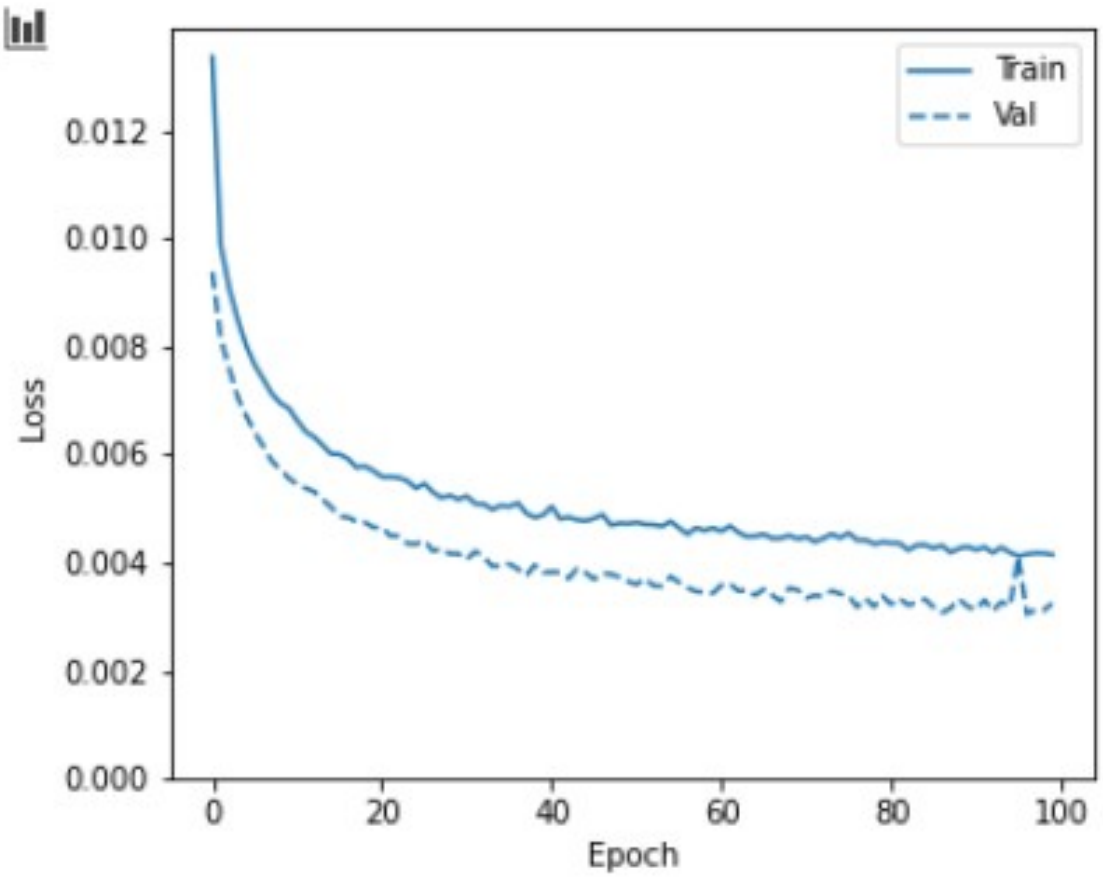


Figure 7.1: The loss-epoch plot of scenario 2 MLP model shows that the training/validation loss has not stabilised and still continues to reduce when the training process stopped. This means that the model is under-fitting.

7.2.1 Summary & Conclusion

In Chapter 6 & Chapter 7, the PaySim transaction dataset has been used to train several machine learning and ANN-based MLP models under various experimentation scenarios to accomplish the following research objective,

Investigate the extent to which ML and ANN-based fraud detection models trained using synthetic transaction data detect financial transaction fraud

Based on the analysis of the model performance metrics, the following insights have been captured,

- Both Shallow ML and ANN models are capable of detecting fraud in financial transaction data with varying performance. The analysis of the performance metrics gives a clear picture of the strengths and weaknesses of each model.
- Boosting tree algorithms such as XGBoost and LightGBM perform significantly better than linear models like Logistic Regression, Linear SVM, and simpler tree-based models like Random Forest in detecting fraudulent transactions.
- LightGBM models have the best performance of all other ML and ANN models from comparing the metrics without taking into account the cost of increased FNR or FPR.

- XGBoost and LightGBM models have very high AUPRC compared to the other models denoting a lesser need for a trade-off between precision and recall. This explains the low false positive and negative rates compared to the other decision-tree ensemble - Random Forest.
- XGBoost has slightly lesser False Negative Rate (FNR) and higher AUPRC than LightGBM models. Given the cost of increased FNR is higher than FPR, the XGBoost model trained under the current experimentation setting is better for the fraud detection use case. Also, XGBoost performs better than LightGBM in generalisability under the current experimentation settings.
- In the experiments, XGBoost is performing better than ANN-based fraud detection models. This is not surprising, as much other research shows that the tree-based ensemble models like XGBoost outperform ANN models on tabular datasets Lundberg et al. [2019] Chen and Guestrin [2016]. This could be explained by the fact the tabular data has no implicit temporal or spatial patterns but the features are "individually meaningful" Lundberg et al. [2019]. ANN models perform better with image and speech data with strong temporal and spatial patterns.
- The ANN models need further hyperparameter optimization as the model is currently underfitting. Furthermore, the scenario 3 & 4 MLP models show that by increasing class representation using oversampling or class-weighting, can indeed decrease the false-negative rate and improve the number of true positives.

8 Explaining Fraud Detection Models: A Proof-of-Concept Demonstration

In Chapter 6, the first phase of research involving the training of ML/ANN-based fraud detection models using the PaySim transaction dataset has been accomplished. In Chapter 7, the trained fraud detection models have been evaluated for their classification performance and their performance results have been summarized. It has been concluded in Section 7.2.1 that both XGBoost and MLP models provide high performance in fraud detection tasks. Among all the models developed for fraud detection, XGBoost with optimal hyperparameters provide the best classification performance in terms of improving the false-negative rate. In the second phase of research, Explainable AI approaches proposed in Chapter 4 have been investigated using the fraud detection models developed in the first phase of research.

In this chapter, two proofs of concepts have been demonstrated for purpose of investigating the feasibility of XAI approaches - *TreeSHAP and Diverse Counterfactuals* in improving the explainability or interpretability of ML/ANN-based fraud detection models. The following two research objectives have been formulated to accomplish this,

- *Demonstrate a proof of concept for improving explainability or interpretability of ML-based fraud detection models using model-agnostic post-hoc explainability approaches*
- *Demonstrate a proof of concept for improving explainability or interpretability of ANN-based fraud detection models using model-agnostic post-hoc explainability approaches*

In Section 8.1, a feature contribution method called TreeSHAP (Refer Section 4.2.3) and in Section 8.2, an example-based method called Diverse Counterfactuals (Refer 4.3.3) have been used for explaining the decision outcomes of ML/ANN-based fraud detection models.

8.1 Feature Contribution Method - TreeSHAP

Tree-based ML models are popular non-linear models that are widely used in medicine, finance, manufacturing, and marketing to make predictions. Interpretability of model behavior and explainability of model outcomes are crucial for enabling trust in these complex black-box models. TreeSHAP is a novel model interpretation tool that aims to provide both global and local explainability to model output. As mentioned in Section 4.2.3, TreeSHAP is a feature contribution method that provides local and global explanations for tree-based models like XGBoost and Random Forest. TreeSHAP algorithm reduces the complexity of computing Shapley values and it is computationally faster than KernelSHAP Lundberg et al. [2019]. It computes Shapley values for any given dataset in polynomial time compared to the much longer exponential time of KernelSHAP. TreeSHAP provides local explanations to model output by assigning Shapley value to each feature of the specific local instance. The Shapley value of each feature gives the relative marginal contribution of each feature to the model prediction. TreeSHAP can also provide global explanations by aggregating the local explanations while "preserving the local faithfulness of the original model" Lundberg et al. [2019]. Refer Section 4.2.3 for a detailed description of TreeSHAP algorithm with the implementation pseudo-code.

8.1.1 Motivation for TreeSHAP

As discussed in Section 3.1.6, model users (fraud detection analysts) require model-agnostic, post-hoc, local, and global explainability to augment their decision-making in the transaction fraud monitoring process 3.1.2. The treeSHAP algorithm can provide model-agnostic, post-hoc, local, and global explainability to any tree-based ML model. In addition to satisfying the above-mentioned explanation characteristics, TreeSHAP provides the following advantages Lundberg et al. [2019],

- **Fast & Optimal Local Explanations:** TreeSHAP computes local explanations in polynomial time. This is significantly faster than KernelSHAP and other feature contribution methods for local explainability like Local interpretable model-agnostic explanations (LIME). LIME provides local explanations by perturbing the samples around the instance of interest and then fitting a linear interpretable model using the perturbed samples and their predictionsRibeiro et al. [2016].
- **Feature Interactions:** "SHAP feature interactions" is a novel feature implemented in TreeSHAP. In addition to the main effects of a feature on a prediction, TreeSHAP can take into consideration the effects of the interactions with other features on a prediction. Refer fig:mortalitymodel for an examples for this unique feature within TreeSHAP.
- **Global Insights:** Advanced implementations of the TreeSHAP algorithm can efficiently aggregate local explanations to provide global insights on the black-box model. They provide a variety of global explanations while maintaining "local faithfulness" to the original model.

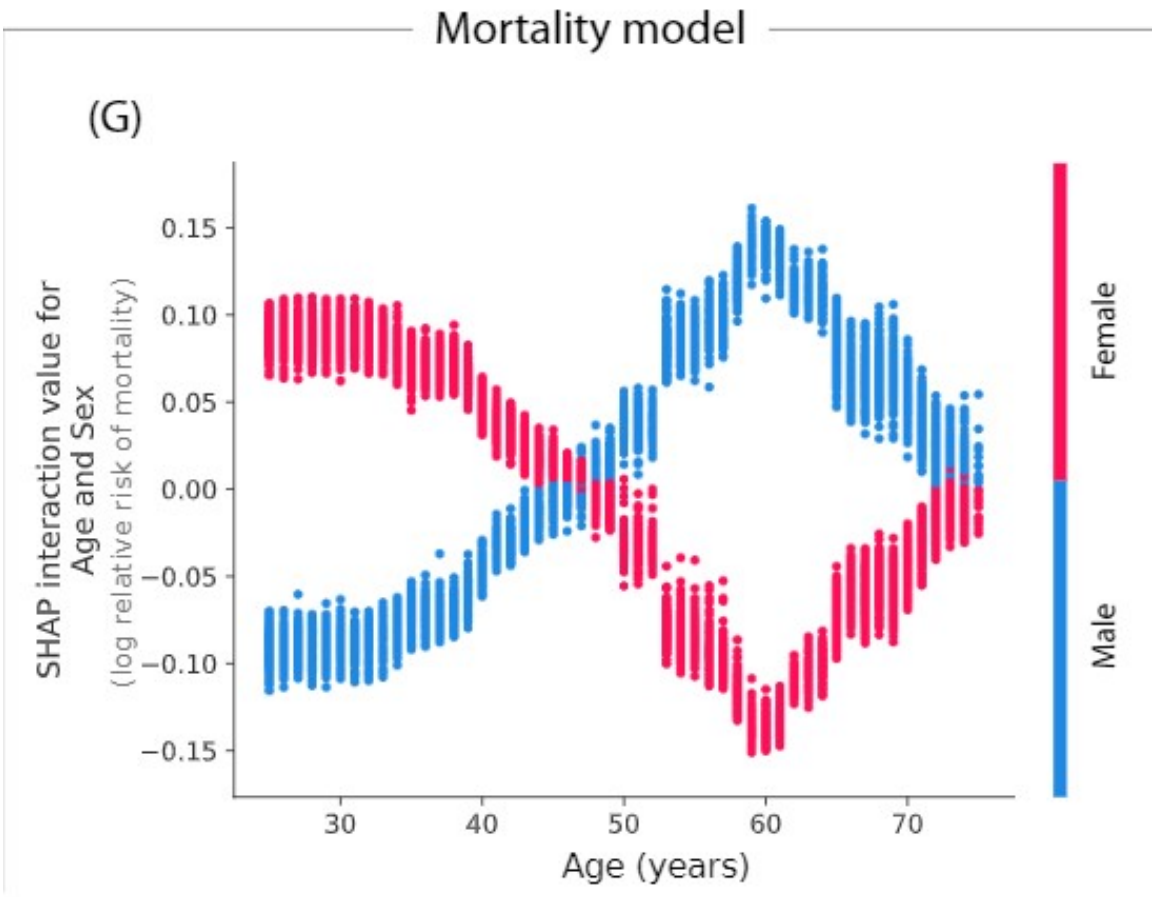


Figure 8.1: This is a feature interaction plot of two features 'sex' and 'age' from a cardio-vascular mortality dataset. The plot shows the "interaction effects" between the two features. The plot shows that the risk of cardiovascular mortality is very high in men compared to women at the age of 60. This information would not be able from a global feature importance plot of either 'age' or 'sex'.

8.1.2 Implementation Details

For implementing model-agnostic post-hoc explainability to fraud detection ML models, the three datasets shown in Table 8.1 have been used to train three different XGBoost models with optimal hyperparameters. The hyperparameters have been tuned using the Bayesian optimization process. The three trained XGBoost models to provide high performance in detecting fraud as shown in Section 7.2.1. TreeSHAP algorithm has been implemented on the three different XGBoost models to generate local and global explanations for model predictions. Various plots have been generated to visualise local and global explanations to model predictions (Refer 8.1.4 & 8.1.3).

Dataset Type	Genuine Transactions	Fraudulent Transactions	Total	Data Skew (%)
Full Dataset	2762196	8213	2770409	0.003
Transfer Dataset	528812	4097	532909	0.008
Cash Out Dataset	2233384	4116	2237500	0.002

Table 8.1: Subsets of preprocessed transaction dataset used to train three different XGBoost-based fraud detection models

8.1.3 Generating Global Explanations using TreeSHAP

Explanations for the global behavior of the fraud-detection ML models can be generated and visualized by implementing the TreeSHAP algorithm. The explanation communication can be visualized through three different plots - *SHAP feature importance plots, Summary plots, and SHAP feature interaction plots.*

SHAP Global Feature Importance Plots

Figure 8.2 is the SHAP feature plot for the XGBoost model trained using the full dataset as shown in Table 8.1. The absolute Shapley values per feature across the full dataset are aggregated. These aggregated Shapley values per feature are in descending order of their absolute magnitudes. This plot shows the feature inputs that have a high contribution to the model predictions. Since Shapley values give the magnitude of feature contributions, it can be seen that the "errorBalanceOrig" feature has a significantly high contribution to the model predictions globally. "errorBalanceOrig" is an artificially engineered feature to account for the data quality issues in the synthetically generated dataset. "errorBalanceOrig" is given by the following equation,

$$errorBalanceOrig = newBalanceOrig + amount - oldBalanceOrig \tag{8.1}$$

It can be seen that the artificial feature is made up of other original features of the dataset. This could mean that "newBalanceOrig", "oldBalanceOrig" and "amount" implicitly contribute to the high feature contribution "errorBalanceOrig". The second and third important features are "oldBalanceDest" and "newBalanceOrig". It is surprising to note that the "errorBalanceDest" feature has no significant contribution to model output. In Exploratory Data Analysis (EDA), this feature linearly separated fraudulent transactions from genuine (Refer Figure 5.3).

Both "type_CASH_OUT" and "type_TRANSFER" features have no significant global importance to model predictions. This means that the type of transaction being either "Cash Out" or "Transfer" has more or less the same contribution to the model output. To investigate further, the SHAP feature importance plots for XGBoost models trained using standalone "Cash Out" and "Transfer" datasets are shown in Figure 8.3 & Figure 8.4. From Section 5.3, it can be noted that the modus operandi of fraud in the PaySim transaction dataset is that the alleged criminal account makes a Transfer transaction from the genuine account to itself and then makes a Cash-Out transaction to a third party. This implies that the criminal account is the originator of a 'Cash Out' transaction and the destination of a 'Transfer' transaction. In Figure 8.3, it can be seen that both "newBalanceDest" and "oldBalanceDest" have high feature

contributions to the model predictions in the Transfer transaction type. It makes sense as the destination accounts of the Transfer transaction type are the criminal account and the features related to the criminal account's balances significantly contribute to the probability of fraud. In Figure 8.4, it can seen that "newBalanceOrig" , "errorBalanceOrig" and "oldBalanceOrig" have high feature contributions to the model predictions in the "Cash Out" transaction type. It makes sense as the originator accounts of the Cash Out transaction type is the criminal account and the features related to the criminal account's balances significantly contribute to the probability of fraud. It should also be noted that the "newBalanceDest" feature and "newBalanceOrig" features have the most impact on the models trained using "Transfer" and Cashout datasets respectively. This could mean that the account balances after a "Transfer" or "Cash Out" fraudulent transaction increases and decreases respectively. The magnitude of this increase and decrease in the criminal account's balance after a fraudulent transaction has to be investigated further with the use of summary plots. The feature importance plot is useful to have a sense of the most contributing features to the global behavior of the model. It does not provide any information regarding how the range of feature values impacts the model globally. This can be visualized using the summary plots.

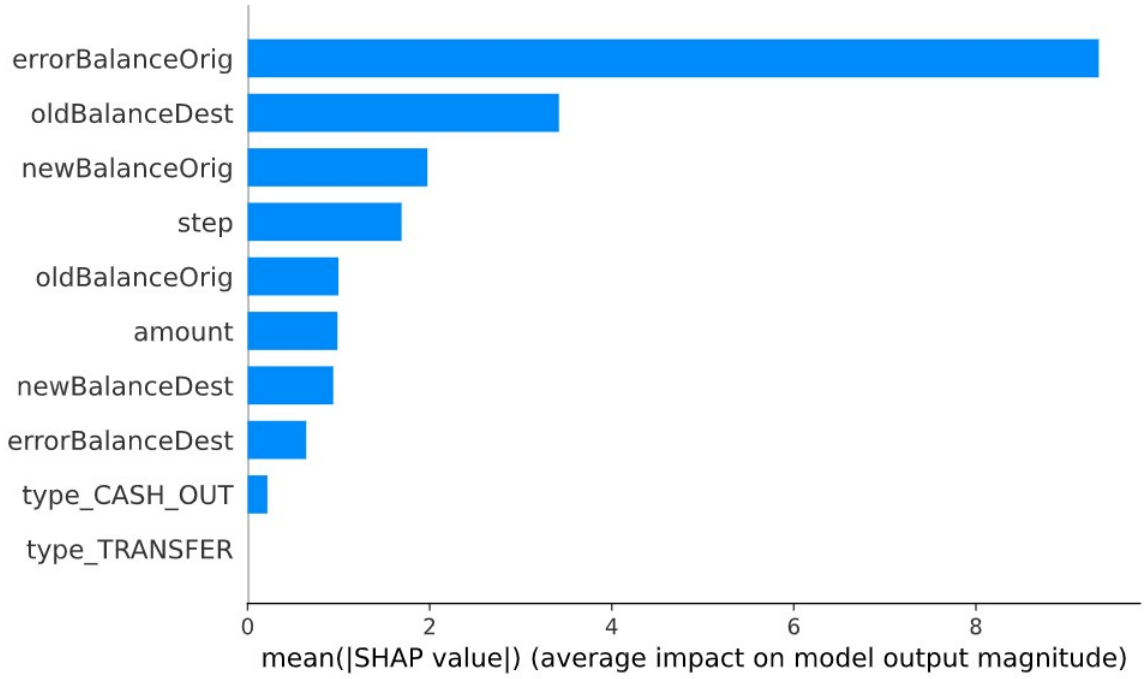


Figure 8.2: Global feature importance of the XGBoost model trained using the full dataset to predict fraud as shown in Table 8.1

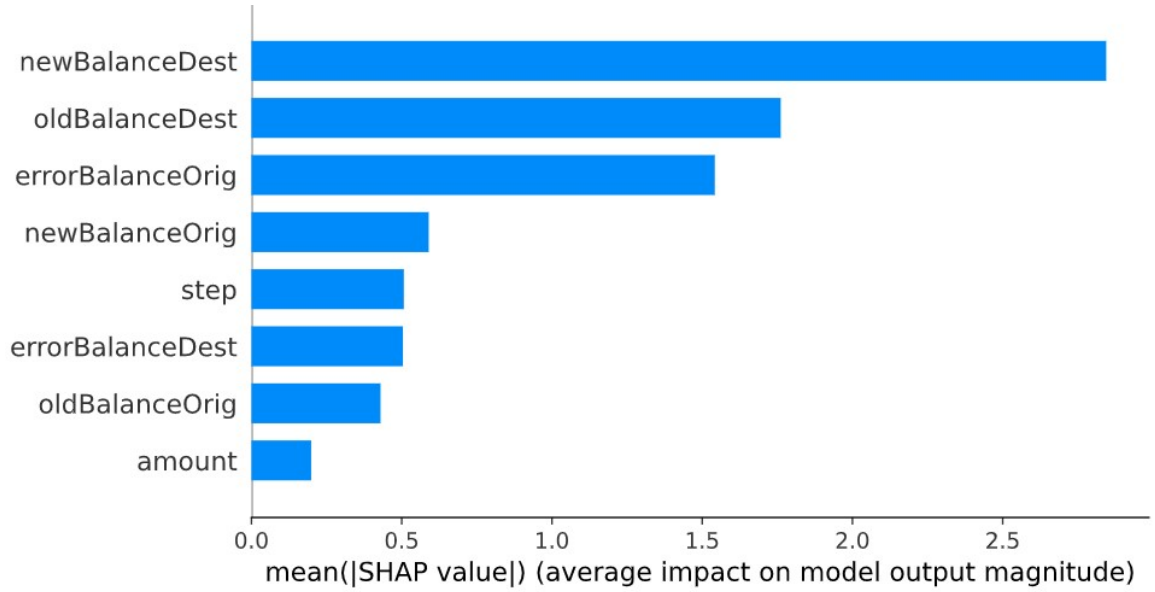


Figure 8.3: Global feature importance of the XGBoost model trained using only the 'Transfer' dataset to predict fraud as shown in Table 8.1

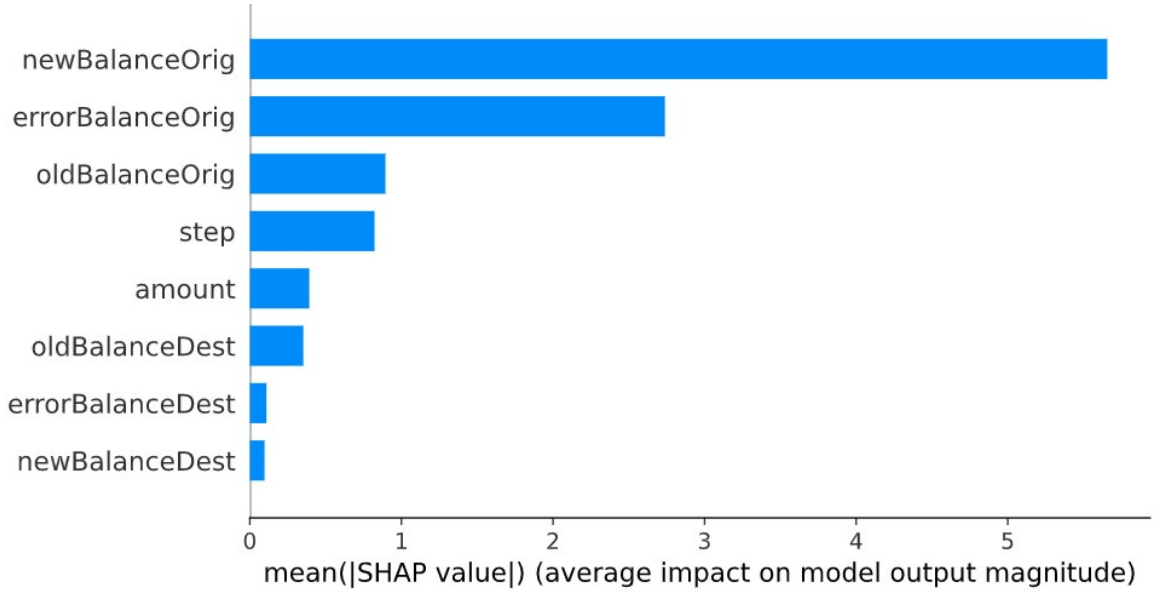


Figure 8.4: Global feature importance of the XGBoost model trained using only the 'Cash Out' dataset to predict fraud as shown in Table 8.1

Summary Plots

The summary plot in Figure 8.5 combines feature effects with feature contributions to the fraud detection model output Lundberg et al. [2019], Lundberg et al. [2018], Molnar [2020]. The summary plot can show the relationship between the range of feature values (Feature effects) and the contribution to model output. The Y-axis gives the feature contributions to the model output and the X-axis gives the SHAP value for each instance. The SHAP value of various instances that overlap at jittered along the

Y-axis. This shows how much the SHAP values vary per feature. The effect of "newBalanceOrig" feature values on the model output is clear from Figure 8.5. For higher values of the "newBalanceOrig" feature, the impact on model output (SHAP value) is very low. This means that when the new balance in the originator account after any transaction (Cash Out or Transfer) is high, the probability of predicting fraud is lower. This could mean that at least in the transaction dataset, a large amount is moved out of the originator account in fraudulent transactions. Looking into the "amount" feature, the probability of predicting fraud increases if the transaction amount is high. Lower transaction amount has a negative impact on the probability of predicting fraud. From "errorBalanceDest", it can be seen that the impact on model prediction is high for a range of high feature values. This means that "errorBalanceDest" is not that straightforward in separating fraudulent from genuine transactions. For high values of the "step" feature, the model impact is lower than for low values. This means that the transactions during the later part of the day have reduced the probability of fraud risk. The summary plots provide a lot more information of the "main" feature effects on the model outcome but it does not take into account the "interaction" effects between two features. To investigate feature interaction effects on the model predictions, SHAP feature interactions plots can be used Lundberg et al. [2019].

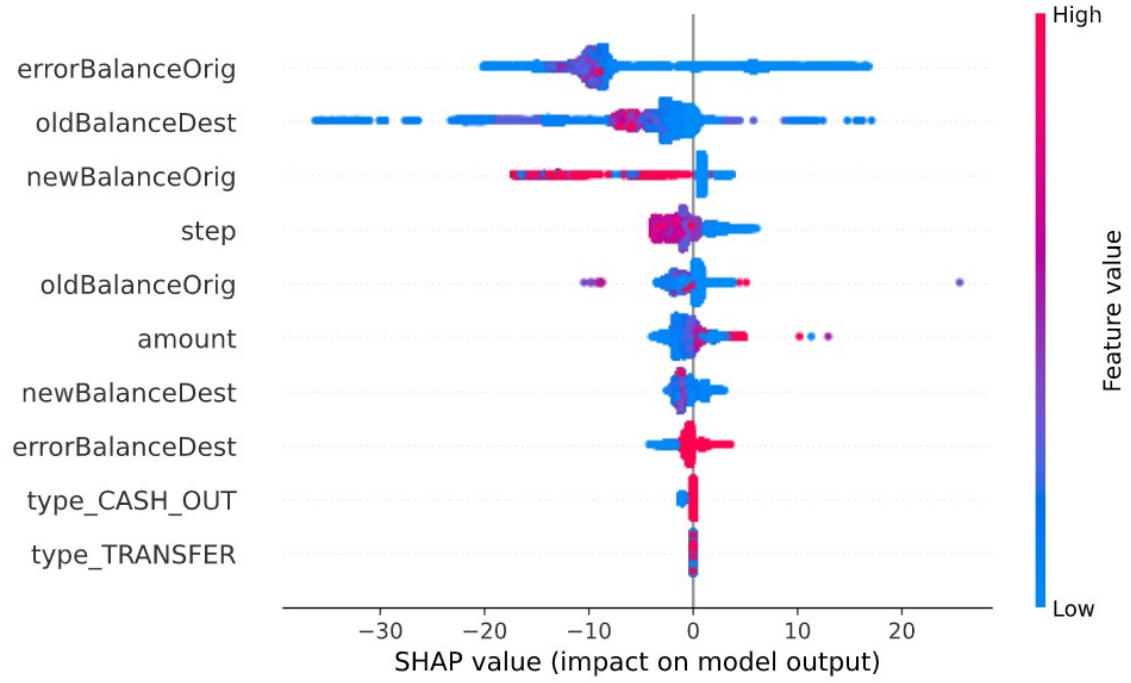


Figure 8.5: Summary plot of the XGBoost model trained using the full dataset as shown in Table 8.1

Feature Interaction Plots

In the previous section, the individual feature effects on the model outcome called "main" feature effects have been visualized. In addition to the "main" feature effects, combined feature effects on the model outcome called "interaction" feature effects have been estimated and visualized using the TreeSHAP algorithm in this section. TreeSHAP estimates *SHAP interaction values* (Refer Section 4.2.3 for more details) to visualize the combined effects of two features on the model outcomes Lundberg et al. [2019]. Figure 8.6 shows the interaction effects of two features - "errorbalanceOrig" and "step" on the model outcome. For the time step ranging from 200 to 300, the impact on the model outcome due to a second feature "errorBalanceOrig" stays the same. On the other hand, the impact on model outcome varies with varying time steps only when the "errorBalanceOrig" feature is zero. It shows that when the "step" feature ranges between 200 to 300, the impact on the model outcome due to the "errorBalanceOrig" feature is insignificant irrespective of its value.

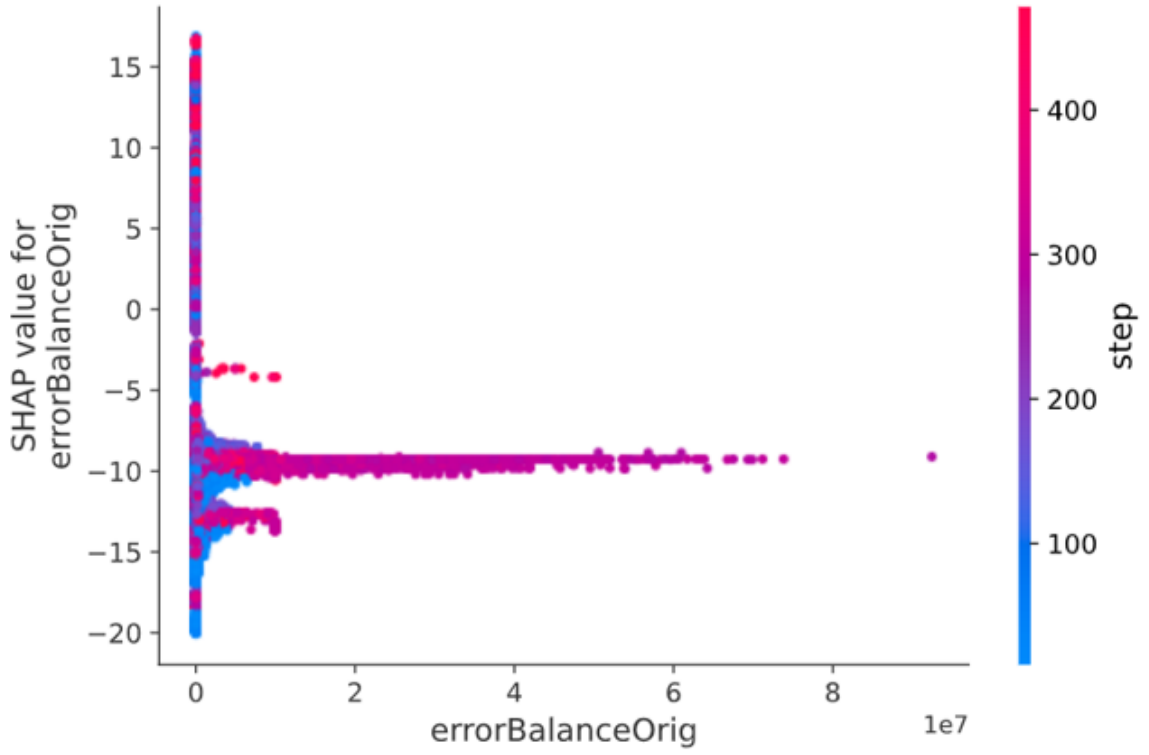


Figure 8.6: **A Feature Interaction Plot showing the interaction effects of two features - "errorbalanceOrig" and "step" on the model outcome.** It shows that when the "step" feature ranges between 200 to 300, the impact on the model outcome due to "errorBalanceOrig" feature is insignificant irrespective of its value.

8.1.4 Generating Local Explanations using TreeSHAP

The global explanations are very useful for model developers who seek to understand the global model behavior and improve its performance Lundberg et al. [2019]. On the other hand, local explanations are much more suitable for model users (fraud detection analysts) who analyze specific transactions based on the Suspicious Activity Report (SAR) (Refer Section 3.1.3). To facilitate local explainability, local feature importance plots or force plots are available in TreeSHAP. Local feature importance gives the feature contributions for a specific model prediction of the model. In the force plot, Shapley-value-based feature contribution scores are represented as "forces" Lundberg et al. [2018], Lundberg et al. [2019], Molnar [2020]. The increase or decrease in the probability of predicting fraud is "forced" by the increase or decrease in feature values for a specific transaction. The increase or decrease in prediction is forced by the feature values from the baseline. The baseline is given the average probability of model prediction of all the transactions in the dataset Lundberg et al. [2019]. For the dataset used, it is calculated to be 1.292. In the force plot for the transaction ID 6362619 (Refer Figure 8.7), the increase or decrease in prediction from the baseline is given by "higher" or "lower" arrowheads. This specific transaction is fraud and is also correctly predicted to be fraud by the model. The "errorBalanceOrig" feature has forced the model prediction to be higher than the baseline prediction and therefore making the model classify it as a fraudulent transaction. In the force plot for the transaction ID 3 (Refer Figure 8.8), the errorBalanceOrig feature has forced the model prediction to be lower than the baseline prediction. This explains why the model classified it as a genuine transaction.

Figure 8.7: Force plot for fraudulent transaction (ID 6362619)

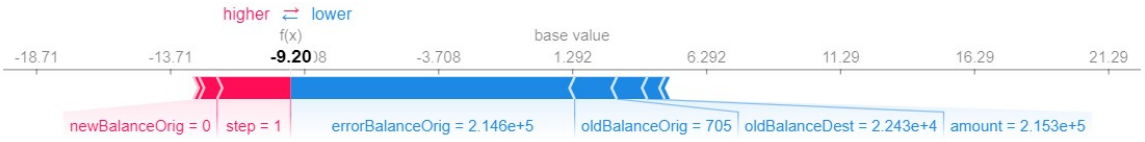


Figure 8.8: Force plot for genuine transaction (ID 3)

8.1.5 Shapash Interactive Dashboard

Figure 8.9 shows the Shapash interactive dashboard developed using the Shapash python package [1] . The so-called Shapash Monitor provides an interactive graphical user interface for generating and visualizing local and global explanations for model predictions. The dashboard can be used by model users like fraud detection analysts who don't have the technical knowledge for applying the TreeSHAP algorithm to the fraud detection models. As shown in Figure 8.9, the dashboard provides tools to visualize the global feature importance plots, local feature importance plots, summary plots, and the ability to analyze individual features interactively.

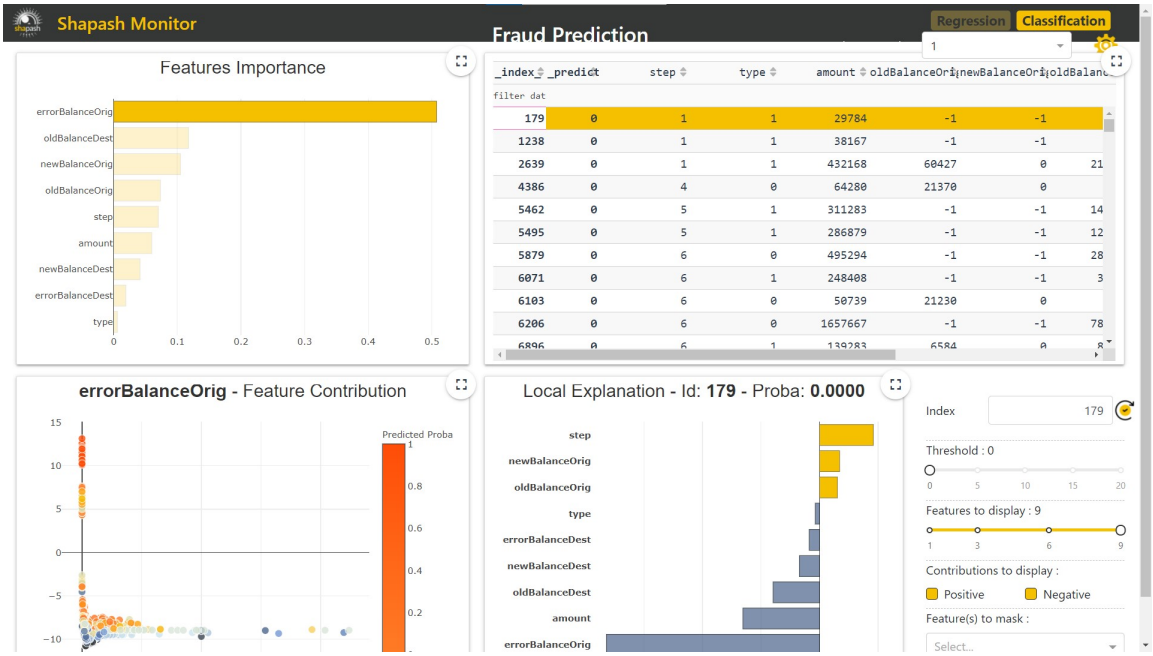


Figure 8.9: Shapash Interactive Dashboard

[1] https://github.com/MAIF/shapash

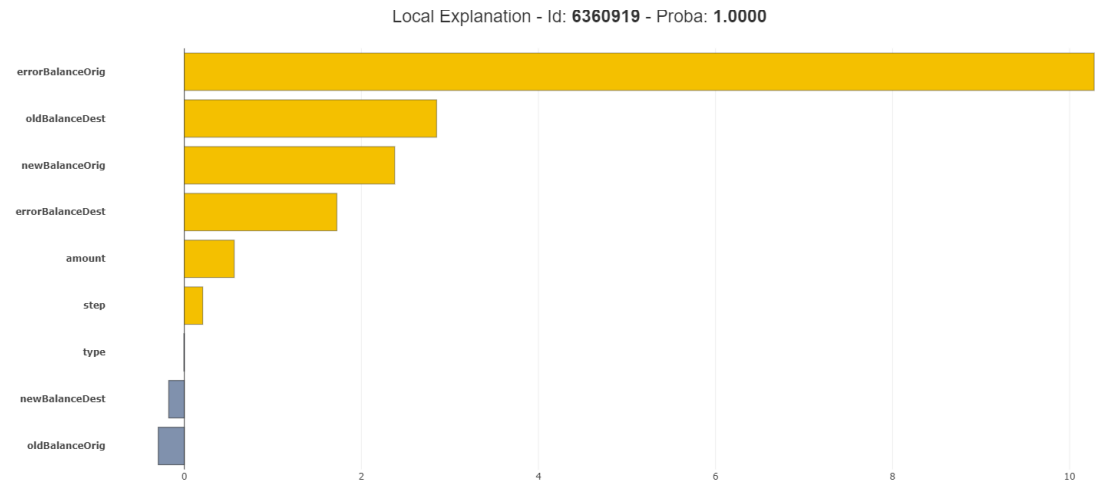


Figure 8.10: Local feature importance for a fraudulent transaction (ID 6360919). errorBalanceOrig feature is significantly contributing to probability of fraud for this transaction

8.2 Example-based Method - Diverse Counterfactuals

8.2.1 Motivation for Diverse Counterfactuals

A counterfactual explanation (CE) can provide a "what-if" analysis of ML/ANN model outcomes to help explain causal relations like "If X had not occurred, Y would not have occurred" Molnar [2020]. The python package DiCE provides a code-level implementation of Diverse Counterfactuals that helps in providing post-hoc 'diverse' and 'feasible' CE to ML and ANN model predictions. A detailed description of Diverse Counterfactuals with their implementation algorithm is given in Section 4.3.3. This XAI approach can be used to generate counterfactual explanations for understanding changes in the feature space that lead to a positive change in the output Mothilal et al. [2020b]. For example, DiCE can help model developers understand the specific set of features that has to change to affect an output change from genuine to fraudulent or vice-versa in an ML/ANN-based fraud detection model. A diverse counterfactual explanation means that for an instance of interest, the counterfactual algorithm should generate multiple unique counterfactual explanations. Feasible counterfactual explanations mean that the generated counterfactuals should have values closer to the instance of interest. The algorithmic definition of 'diverse' and 'feasible' CE is given in Section 4.3.3. DiCE can provide simple and clear explanations to the decision boundary of the ML/ANN-based fraud detection models. *The fraud analyst who analyses the fraudulent transactions with the help of Suspicious Activity Reports (SAR) can have a better understanding of how and which feature input changes contribute to a fraud prediction and thereby making them better decision-makers*. The main advantage of applying DiCE to fraud detection models is that the explanations are business intuitive and very human-friendly for non-technical stakeholders like fraud detection analysts. Unlike feature contribution methods like TreeSHAP, DiCE can provide explanations in the same form and nature of model output itself.

Much recent research in the domain of CE hints at the possibility of applying the counterfactual approach to explaining fraud detection models Hashemi and Fathi [2020], Blanchart [2021], Cartella et al. [2021]. But there is no research available on the investigation of 'diverse' and 'feasible' counterfactuals to explain the outcomes in fraud detection models. Diverse counterfactuals explanations to fraud detection models can help fraud detection analysts to hypothesis the different ways fraud can be committed. Feasible counterfactual explanations can help fraud detection analysts with realistic explanations of fraudulent outcomes. For example, customer ID in a transaction dataset should not be changed by the algorithm to provide a counterfactual explanation as customer ID is unique to each customer. Moreover, user-defined context can be enforced on the counterfactual explanations to restrict feature changes completely or within a range of values, Mothilal et al. [2020b]. Therefore, by restricting feature way

changes of certain sensitive features like race, ethnicity, and area post-code, bias in decision-making can be mitigated. By presenting the fraud detection analysts with counterfactual explanations that don't incorporate the changes in sensitive features like race or ethnicity, discrimination in AI-driven decision-making can be avoided.

This thesis research is pioneering a proof of concept demonstration to investigate diverse and feasible counterfactuals for improving the explainability or interpretability of ML/ANN-based fraud detection models.

8.2.2 Implementation Details

Highly performing XGBoost (ML) and MLP-based (ANN) fraud detection models that have been developed in Chapter 6 have been used for demonstrating the proofs of concepts in this section. The proofs of concepts involves generating "diverse" and "feasible" counterfactual explanations for genuine transactions and fraudulent transactions. Diverse counterfactual explanations for XGBoost model outcomes have been discussed in Section 8.2.4. Diverse counterfactual explanations for MLP model outcomes have been discussed in Section 8.2.5. The generated counterfactual explanations are visualized in a Pandas data-frame format.

8.2.3 Generating Diverse Counterfactuals

DiCE is a python library that provides code-level implementations of various 'diverse' and 'feasible' counterfactual algorithms. DiCE is still under development and therefore lacks stability when it comes to many of its features. It requires a bit of an effort to be able to use it successfully for generating both diverse and feasible counterfactuals. The DiCE tool provides three different model agnostic methods to generate diverse counterfactuals Mothilal et al. [2020b] with varying functionality. The three methods are as follows,

- **Random Sampling**: In this method, an initial counterfactual is randomly sampled, and then it is used along with an instance of interest to minimize a loss function. The local optimum of the loss function gives the required counterfactual instance. Refer Algorithm 8 for an algorithmic implementation pseudo-code for this counterfactual method.
- **Genetic Algorithm**: Schleich et al. [2021] proposes a novel genetic algorithm-based method for generating feasible counterfactual explanations in real-time. It is called "GeCo" and it is the most state-of-art algorithm in the class of counterfactual approaches.
- **KD Tree Algorithm:** Van Looveren and Klaise [2019] proposed an improved, faster model agnostic method to generate interpretable counterfactual examples for classification tasks by using class prototypes. Refer Algorithm 7 for an algorithmic implementation pseudo-code for this counterfactual method.

8.2.4 Generating Diverse Counterfactuals for Genuine Transactions

In Figure 5.3, it is shown that the "errorBalanceDest" feature distinctly separates genuine transactions from fraudulent transactions. The feature input change from "-errorBalanceDest" to "+errorBalanceDest" causes the model prediction to switch from genuine to fraudulent transactions. The generated counterfactual proves that the XGBoost model has captured this specific characteristic of the "errorBalanceDest" feature. As shown in Figure 8.11, the amount feature has to change from EUR 13,717 to EUR 14,025,256 for the prediction to switch to fraudulent. Since it is a significant value change, it could not be considered as a feasible counterfactual explanation. It is now interesting to investigate if the counterfactual explanation (genuine to fraudulent) due to feature input change from "-errorBalanceDest" to "+errorBalanceDest" is seen in other transaction instances as well. For this, a transaction of ID 3892633 has been considered for generating ten different diverse counterfactual explanations.

Query instance (original outcome : 0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	44	1	13717.53	19207.0	5489.47	3251100.68	3268360.52	0.0	-3542.31	0

Diverse Counterfactual set (new outcome: 1.0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	456.0	-	14025256.6	-	-	-	-	-	4400269.5	1.0
1	-	-	39386918.2	-	44164337.6	-	-	-	-18047822.0	1.0

Figure 8.11: Diverse counterfactual explanations trying to answer the question **"What feature changes can cause a switch from genuine to fraudulent transaction?"** - It can be noted that a feature change from "-errorBalanceDest" to "+errorBalanceDest" causes the model prediction to switch from genuine to fraudulent transaction.

Figure 8.12 shows the ten different counterfactuals generated for a specific transaction instance of ID 3892633. It is noted that the feature input change from "-errorBalanceDest" to "+errorBalanceDest" does not influence the outcome switch from genuine to fraudulent for this specific transaction. The counterfactuals, in this case, have not captured the pattern for fraudulent transactions embedded in the synthetic transaction dataset. It could be due to the following reasons,

- During the optimization of the counterfactual loss function (Refer Equation 4.16), the algorithm could have found that the feature input changes in "amount" and "oldBalanceOrig" had a significant impact on the reduction of counterfactual loss than "errorBalanceDest" feature for this specific transaction.
- This could be an outlier case where the XGBoost model has not properly captured the pattern for fraudulent transactions embedded in the synthetic transaction dataset. From the model results (Refer Table 7.1). the model is not 100% efficient in capturing the nuances in the synthetic transaction dataset.

... Query instance (original outcome : 0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	284	CASH_OUT	307041.40625	503646.0625	196604.625	775812.125	1164464.25	-0.01	-81610.71875	0

Diverse Counterfactual set (new outcome: 1.0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	-	-	45716987.4	17457824.5	-	-	-	-	-	1
1	-	-	71514895.3	55535523.4	-	-	-	-	-	1
2	-	-	16809471.5	57074952.4	-	-	-	-	-	1
3	-	-	20440183.3	46461565.2	-	-	-	-	-	1
4	723.0	-	15249420.6	-	-	-	-	-	-	1
5	-	-	15159170.6	47872579.7	-	-	-	-	-	1
6	-	-	40617037.7	15290203.4	-	-	-	-	-	1
7	735.0	-	-	-	-	-	-	-	-	1
8	732.0	-	-	-	-	-	-	-	-	1
9	-	-	9916465.6	24542867.8	-	-	-	-	-	1

Figure 8.12: **Ten different counterfactual explanations have been generated for the transaction ID 3892633**. It can be seen that the feature input change from "-errorBalanceDest" to "+errorBalanceDest" does not influence the outcome switch from genuine to fraudulent for this specific transaction.

User-defined Constraints

In DiCE, constraints can be put on the feature input changes to incorporate user-defined context to the generation of counterfactual explanations. Using the parameter "features_to_vary" in the method gen-

erate_counterfactuals(), a constraint can be imposed on which features can change to affect an outcome switch from genuine to fraudulent. When the algorithm had been forced to generate counterfactual explanations by changing the feature inputs of only one feature "errorBalanceDest", it failed to generate any counterfactual explanation for transaction ID 3892633. It makes sense as it is unlikely for counterfactuals to exist for a specific transaction when input changes happen in only one feature. Subsequently. the following features - 'errorBalanceOrig','oldBalanceDest','newBalanceOrig','step','oldBalanceOrig','amount' had been allowed to value but with range constraints on these features. In DiCE, constraints can be imposed on the feature ranges to generate counterfactual explanations. For example, the transaction amount can be constrained to generate feasible counterfactual explanations, For this, the parameter "permitted_range" parameter is used. The "amount" and "oldBalanceOrig" features had been constrained to change in the range of EUR (200000, 1000000) and EUR (300000, 2000000) respectively. As shown Figure 8.13, only three counterfactuals could be generated under the specific constraints. It can be concluded that diverse counterfactuals under user=defined constraints do not always exist and the user has to experiment with different user-defined constraints to increase the chances of generating diverse counterfactual explanations for a specific transaction instance.

Query instance (original outcome : 0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	284	CASH_OUT	307041.40625	503646.0625	196604.625	775812.125	1164464.25	-0.01	-81610.71875	0

Diverse Counterfactual set (new outcome: 1.0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	729.0	-	-	-	-	-	-	-	-	1
1	729.0	-	307041.49999676837	-	-	-	-	-	-	1
2	730.0	-	-	-	-	-	-	-	-	1
3	732.0	-	-	-	-	-	-	-	-	1
4	743.0	-	-	-	-	-	-	-	-	1
5	737.0	-	-	-	-	-	-	-	-	1
6	735.0	-	-	-	-	-	-	-	-	1
7	736.0	-	-	-	-	-	-	-	-	1
8	736.0	-	-	-	32620996.0	-	-	-	-	1
9	743.0	-	-	-	23159902.6	-	-	-	-	1

Figure 8.13: Ten different counterfactual explanations have been generated for the transaction ID 3892633 by constraining which features can change and how much they can change

8.2.5 Generating Diverse Counterfactuals for Fraudulent Transactions

In the previous section, diverse counterfactual explanations had been generated for outcome switch from genuine to fraudulent. Subsequently, in this section, several diverse counterfactual explanations had been generated for the outcome switch from fraudulent to genuine. In this experiment, a fraudulent transaction of ID 6112506 had been selected for generating diverse counterfactuals. Figure 8.14 shows an overview of the multiple diverse counterfactual explanations generated for transaction ID 6112506. The feature input change from "+errorBalanceDest" to "-errorBalanceDest" causes the model prediction to switch from fraudulent to a genuine transaction. The generated counterfactual proves that the XGBoost model has captured this specific characteristic of the "errorBalanceDest" feature. Since constraints have not been imposed on the feature input changes and the feature ranges, there have been no missing counterfactual explanations. Although, the generated diverse counterfactuals could not be considered feasible due to the significant changes in the feature input values.

Query instance (original outcome : 1)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	526	CASH_OUT	319883.28125	319883.28125	0.0	102882.007812	422765.28125	0.0	5.820766e-11	1

Diverse Counterfactual set (new outcome: 0.0)

	step	type	amount	oldBalanceOrig	newBalanceOrig	oldBalanceDest	newBalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	-	-	-	-	-	-	-	3729769.0	19596.6	0
1	-	-	-	-	-	-	-	70255596.5	-	0
2	-	-	-	-	-	-	-	90207384.0	-	0
3	-	-	-	-	-	-	-	92127101.6	-48847046.1	0
4	-	TRANSFER	-	43009284.0	-	-	-	-	-	0
5	-	-	-	42501046.0	-	-	-	-	-25730817.6	0
6	-	-	-	14165106.8	-	117162834.1	-	-	-	0
7	457.0	-	-	-	35203843.3	-	-	-	-	0
8	-	-	44522527.5	-	-	-	-	25297477.4	-	0
9	-	-	-	-	16791931.0	-	-	-	-	0
10	-	-	-	42902967.0	-	-	355364609.4	-	-	0
11	-	-	-	-	18927344.4	-	-	-	-16059562.3	0

Figure 8.14: Multiple different counterfactual explanations have been generated for the transaction ID 6112506 - The feature input change from ”+errorBalanceDest” to ”-errorBalanceDest” causes the model prediction to switch from fraudulent to genuine transaction

8.2.6 Estimating local and global feature contribution using Diverse Counterfactuals

Mothilal et al. [2020a] suggests that local and global feature contribution scores can be estimated by generating diverse counterfactual explanations. Feature contribution scores are estimated by identifying the features that change the most to cause an outcome switch. To investigate this, fifty diverse counterfactual explanations have been generated for fraudulent transaction ID 6112506. It has been done to investigate which specific feature has to change the most throughout the fifty counterfactual explanations to cause an outcome switch from fraudulent to genuine. Depending on how frequent a specific feature changes throughout the fifty counterfactual explanations, local feature contribution scores have been estimated as shown in Figure 8.15. For the transaction ID 6112506, ”oldbalanceOrig” has the highest feature importance score of 0.6, denoting that this feature has the high contribution to this specific model outcome. Thereby, it also means that ”oldbalanceOrig” feature changes the most to cause a outcome switch from fraudulent to genuine for this specific transaction instance.

```
[{'oldBalanceOrig': 0.6, 'newBalanceOrig': 0.3, 'errorBalanceOrig': 0.2, 'errorBalanceDest': 0.2, 'newBalanceDest': 0.1, 'type': 0.0, 'step': 0.0, 'amount': 0.0, 'oldBalanceDest': 0.0}]
```

Figure 8.15: An Overview of local feature contribution scores for transaction ID 6112506.

9 Conclusion & Reflection

This thesis research tackled the interpretability problem in Machine Learning (ML) & Artificial Neural Network (ANN)-based fraud detection systems by demonstrating proofs of concepts in Explainable AI. The European Union has been progressing quickly towards regulating high-risk AI applications in its member states through the release of a proposal for an AI act. The Dutch financial regulators - DNB & AFM have provided general guidelines for the financial institutions to improve model explainability or interpretability of their high-risk AI applications. Therefore, the Dutch financial institution sponsoring this thesis research wants to improve the model explainability & interpretability of their high-risk fraud detection systems to be future-proof in terms of legal and regulatory compliance. Hence, the main research objective of the thesis has been,

> To investigate local and global, model-agnostic post-hoc explainability as a proof-of-concept for improving explainability or interpretability in Machine Learning and Artificial Neural Network (ANN) models used for online transaction fraud detection

The main research objective was accomplished through the formulation of four research objectives. In this chapter, the main conclusions have been summarised in Section 9.1 and the limitations & future work has been discussed in Section 9.5. Also, the reflection on academic and socio-technical contributions of this research has been discussed in Section 9.2 & Section 9.3 respectively. Finally, the recommendations to the financial institution sponsoring this research have been discussed in Section 9.4.

9.1 Conclusion

The following are the results from accomplishing the four research objectives,

Literature Study & Selecting XAI Approaches

Research Objective 1: *Describe the state-of-art XAI approaches useful for improving the model-agnostic post-hoc explainability of Machine Learning and Artificial Neural Network (ANN)-based fraud detection models*

In Chapter 4, 40 scientific papers in the domain of Explainable AI have been reviewed using the backward snowballing approach. Among the papers reviewed, and XAI selection criteria as described in Section 4.1 has been used to select and describe the state of art XAI approaches for improving the model-agnostic post-hoc explainability of ML and Artificial Neural Network (ANN) models A detailed description of the selected approaches along with their implementation pseudo-code has been provided. In the end, the approaches have been compared and contrasted in Table 4.1 to understand their relative strengths and weaknesses. Based on the comparative analysis of the selected XAI approaches, feature contribution method - TreeSHAP and example-based method - Diverse Counterfactuals have been chosen to demonstrate the proofs of concepts for the investigation of model agnostic post hoc explainability in ML and ANN-based fraud detection models.

Model Training and Evaluation

Research Objective 2: *Investigate the extent to which ML and ANN-based fraud detection models trained using synthetic transaction data detect financial transaction fraud* In Chapter 5, a detailed Exploratory Data Analysis (EDA) of the synthetic transaction dataset generated by the PaySim data simulator has been conducted. Feature engineering has been performed to mitigate the data quality issues in the dataset. Moreover, an in-depth EDA led to uncovering the pattern of fraud embedded within the synthetic dataset. In Chapter 6, several ML and ANN-based fraud detection models have been trained and evaluating using the transaction dataset generated in Chapter 5. Several experimentation scenarios have been conducted to compare the performance of these models using various evaluation metrics - AUPRC, Precision, Recall, K-fold Cross-Validation. The experimentations involved hyperparameter optimization, SMOTE oversampling, and class-weighting. Hyperparameter optimization has been performed to find the most optimal ML/ANN model for the transaction dataset. While, SMOTE oversampling, and class-weighting has been performed to mitigate model performance issues due to the high class-imbalanced nature of the transaction dataset. Among the ML models, XGBoost & LightGBM classifiers have been the best performing for the use-case at hand. While MLP with optimal hyperparameters performed better than MLP models with SMOTE oversampling and class-weighting. Finally, XGBoost models performed better than MLP models under every experimentation scenario. By investigating the XAI approaches (TreeSHAP & Diverse Counterfactuals), the goal has been to improve model explainability or interpretability of these optimally performing ML and ANN-based fraud detection models in Chapter 8.

Demonstrating Proofs-of-Concept of Explainable AI

Research Objective 3: *Demonstrate a proof of concept for improving explainability or interpretability of ML-based fraud detection models using model-agnostic post-hoc explainability approaches*

The research objective has been accomplished by investigating the TreeSHAP algorithm to improve the model explainability of an XGBoost-based fraud detection model. In Chapter 4, TreeSHAP has been used to improve model-agnostic global and local explainability by visualizing four different plots - SHAP Feature Importance Plot, Summary Plot, SHAP Feature Interaction Plot, and Force Plots. Through these plots, the decision boundary of the models has been well illustrated. Moreover, the plots enabled by TreeSHAP provided in-depth insights into the individual and combined feature effects on the model outcome. Finally, the Shapash python package has been used to develop an interactive dashboard for the visualization of the above-mentioned four plots. An interactive dashboard can help provide non-technical stakeholders like fraud detection analysts with easy-to-access explanations to support their decision-making tasks.

Research Objective 4: *Demonstrate a proof of concept for improving explainability or interpretability of ANN-based fraud detection models using model-agnostic post-hoc explainability approaches*

The research objective has been accomplished by investigating the Diverse Counterfactual algorithm to improve model explainability of XGBoost & MLP-based fraud detection models. In Chapter 4, the DiCE python package has been used to generate counterfactual explanations to genuine and fraudulent transactions. Diverse counterfactual explanations have been generated to explain a model outcome switch from genuine to fraudulent and vice-versa. The algorithm struggled to generate feasible counterfactuals within the desired range of feature values. When user-defined constraints were imposed on the counterfactual optimization process, the algorithm yielded missing or no counterfactual explanations. Several reasons for missing counterfactual explanations under user-defined contexts have been hypothesized. Moreover, depending on the number of times a specific feature changes to cause a switch in the model outcome, the features were provided with a feature importance score. Therefore, it has presented that local feature contribution scores for a specific transaction can be estimated from the summary of counterfactual explanations.

9.2 Academic Reflection

- A review of the key XAI terminologies, concepts, and definitions have helped in mapping out the extensive landscape of XAI. Moreover, XAI selection criteria and comparative analysis of various XAI approaches helped in selecting suitable XAI approaches for the use case at hand. Therefore, the thesis research proposes that comparative analysis of XAI approaches as a qualitative tool for XAI researchers and model developers in identifying the most suitable XAI approach for their use case.

- A review of several existing ML/ANN techniques for fraud detection has been conducted to understand the current challenges in developing ML/ANN-based fraud detection systems. This thesis research proposes the use of synthetic data to overcome the lack of real-world transaction data for research purposes. The research successfully shows the applicability of synthetic data for training and evaluating highly performing ML/ANN-based fraud detection models.

- This thesis research developed several ML and ANN models to compare their performance in classifying genuine from fraudulent transactions. Moreover, the performance of ML/ANN models has been compared to each other but also within the same class of models. For example, multiple optimal XGBoost models with different hyperparameters have been compared to each other.

- This thesis research also proposes the use state of art hyperparameter optimization algorithms such as Bayesian Optimisation and Hyperopt for developing high-performing XGBoost and MLP models respectively.

- This thesis research investigated the impact of the high class-imbalanced nature of the transaction datasets on the performance of ML/ANN-based fraud detection systems. By conducting various experimentation scenarios by combining hyperparameter optimization, SMOTE oversampling, and class-weighting, this thesis provides academic insights into the research gaps in handling class-imbalanced datasets.

- From this thesis research, it has been demonstrated through practical experimentations that XAI approaches can indeed improve explainability in ML/ANN-based fraud detection systems. Feature contribution method - TreeSHAP provides rich insights into the feature effects of ML fraud detection models. This research encourages model developers to use feature contribution methods like TreeSHAP early in the model development process as they can help with debugging their models.

- The proofs of concepts show that model developers can learn more about the model behavior through global explanations like SHAP feature importance plot while model users can become better decision-makers through local explanations such as counterfactuals.

- Finally, this thesis research pioneers the investigation of Diverse Counterfactuals for explaining model outcomes in ML/ANN-based fraud detection systems, This is one of the first such research to explore diverse counterfactuals to help support model users such as fraud detection analysts.

9.3 Socio-Technical Reflection

As demonstrated by the proofs of concepts, XAI can be a powerful tool to improve the model explainability of high-risk AI applications like fraud detection systems in FinTech. Owing to the upcoming EU regulations on AI, XAI can enable a path forward for Fin-Techs to safely adopt black-box AI applications. In addition to helping with regulatory compliance, XAI can enable trust with the model users and the customers as well. XAI can help in identifying discrimination automated into our AI-driven technologies in FinTech, healthcare, and policing.

9.4 Recommendations

Policy Makers:

The EU proposal for a new AI act does not elaborate on what they mean by ”interpretation” tools aia [2021]. The proposal acknowledges the need for financial institutions to mitigate the black-box problem in their AI application but falls short of providing a regulatory framework or a set of guidelines for improving the interpretability of these black-box models. Therefore, the EU and the DNB must share knowledge on the ”interpretation” tools along with a set of high-level guiding principles for the adoption of such interpretation methods. Unless a coherent strategy comes from the policymakers, the practical adoption of XAI or interpretation approaches in financial institutions will lag behind academia. Meanwhile, model developers and users within the financial institutions should conduct collaborative research with the academic to innovate and acquire knowledge regarding the practical adoption of XAI approaches.

Explanation Generation & Communication in Transaction Fraud Monitoring Process

As shown in Figure 9.1, explanation generation and communication should be incorporated in the transaction fraud monitoring process. The Suspicious Activity Report (SAR) used by fraud detection analysts for investigating instances of a fraudulent transaction should be augmented with model explanations along with the historical data on the entities involved in the fraudulent transactions. The visualization of explanations to model prediction should be made interactive for an intuitive communication of model explanations (Similar to Section 8.1.5).

Model Developers

Feature contribution approaches like TreeSHAP provides explanations by creating approximations for the behavior of the original model. TreeSHAP has limitations with model truthfulness. It gives only feature “main” effects and “interaction” effects on the model outcome. TreeSHAP could not be intuitive for non-technical stakeholders like fraud detection analysts who may not be aware of the technical details of the data and the ML/ANN model. On the other hand, TreeSHAP provides rich insights into model behavior both globally and locally. Model developers should make use of TreeSHAP during the development process as it can provide great insights into the strengths and weaknesses of pre=production models. TreeSHAP should also be used to investigate if the model behavior of productionalized models changes over time. Therefore, TreeSHAP should be incorporating in the Continuous Integration/Continuous Development (CI/CD) development pipelines to be able to generate explanations during model development. Also, TreeSHAP should be incorporated as an interactive dashboard in a Machine Learning Operations (MLOps) environment to generate explanations when the model is in production and handling real-world data.

Model Users

Diverse Counterfactuals give explanations that are identical to model output. They are truthful to the original model. Diverse Counterfactuals are easy to understand by non-technical stakeholders due to the simple nature of the explanations generated. Therefore, model users like fraud detection analysts should be provided with counterfactual explanations to specific fraudulent transactions to augment their decision-making process.

9.5 Limitations & Future Work

Several limitations have been encountered during the research and these limitations inform the future work to be done.

- **Data:** The PaySim synthetic transaction dataset is not representative of real-world financial transactions. PaySim dataset represents transactions between accounts within the same bank while real-world transactions usually occur between banks originating from different countries. The transaction flow is not always one-to-one, real-world transactions usually have an interconnected graph structure. Criminals and money launderers usually create complex transaction flows for committing fraud. They do this to hide their identity and evade detection by fraud detection systems. Moreover, the pattern of fraud in the real world is dynamic and keeps evolving. Criminals come up with new types of fraud and the fraud detection models should be continuously re-developed to detect the new types of fraud. To simulate complex transaction flows that evolve, IBM's synthetic transaction generator called AMLSim [1] can be used for further advanced studies. It provides increased functionalities for synthetically generating complex transaction flows to research sophisticated fraud detection systems.
- **Evaluation of XAI approaches:** Evaluation of XAI approaches has not been undertaken in this research. Evaluating explainability approaches is an under-researched domain with a lot of potentials. It lies in the intersection of human-computer interactions, cognitive sciences, and sociology, and therefore, it demands a holistic interdisciplinary research approach. As explanations to ML models are to be presented to humans for comprehension, human-grounded evaluations of the XAI approaches are more important than purely quantitative evaluation metrics for algorithmic efficiency Sanneman and Shah [2020]. Future research investigating XAI approaches should include human-grounded evaluation of the approaches to understanding human comprehensibility.

  Sanneman and Shah [2020] proposes the following human-grounded evaluation of any XAI approach,

  - The model users should be asked to label model outputs into different classes depending on the presented local feature importance plots.
  - Survey-like questions should be asked to find if the users can understand the model behavior from the explanations communicated to them. This is similar to the "goodness" of the explanation scale mentioned in Hoffman et al. [2018].
  - As suggested by Doshi-Velez and Kim [2017] human-grounded evaluation involving "counterfactual simulation" of model behavior for different inputs should be carried out. Counterfactual simulations should be very human-friendly and easier to evaluate, compared to Shapley value-based methods

  .
- **Diverse Counterfactual Explanations:** Algorithms trying to generate diverse and feasible counterfactuals take a long time to generate explanations due to the low computational efficiency of these algorithms. Due to the interdependence between the features in the transaction dataset, feasible counterfactuals are harder to obtain. To improve computational efficiency and generate counterfactual explanations under under-defined constraints, the genetic algorithm-based method proposed by Schleich et al. [2021] should be investigated. The genetic algorithm-based "GeCo" method proposes to generate counterfactuals in real-time. If true, real-time counterfactual explanations can greatly help fraud detection analysts to quickly investigate and decide on the nature of a financial transaction. Thereby, benefiting the customers, reducing the manpower spent on decision-making, and quickly identifying newly emerging patterns of fraud.

[1] https://github.com/IBM/AMLSim

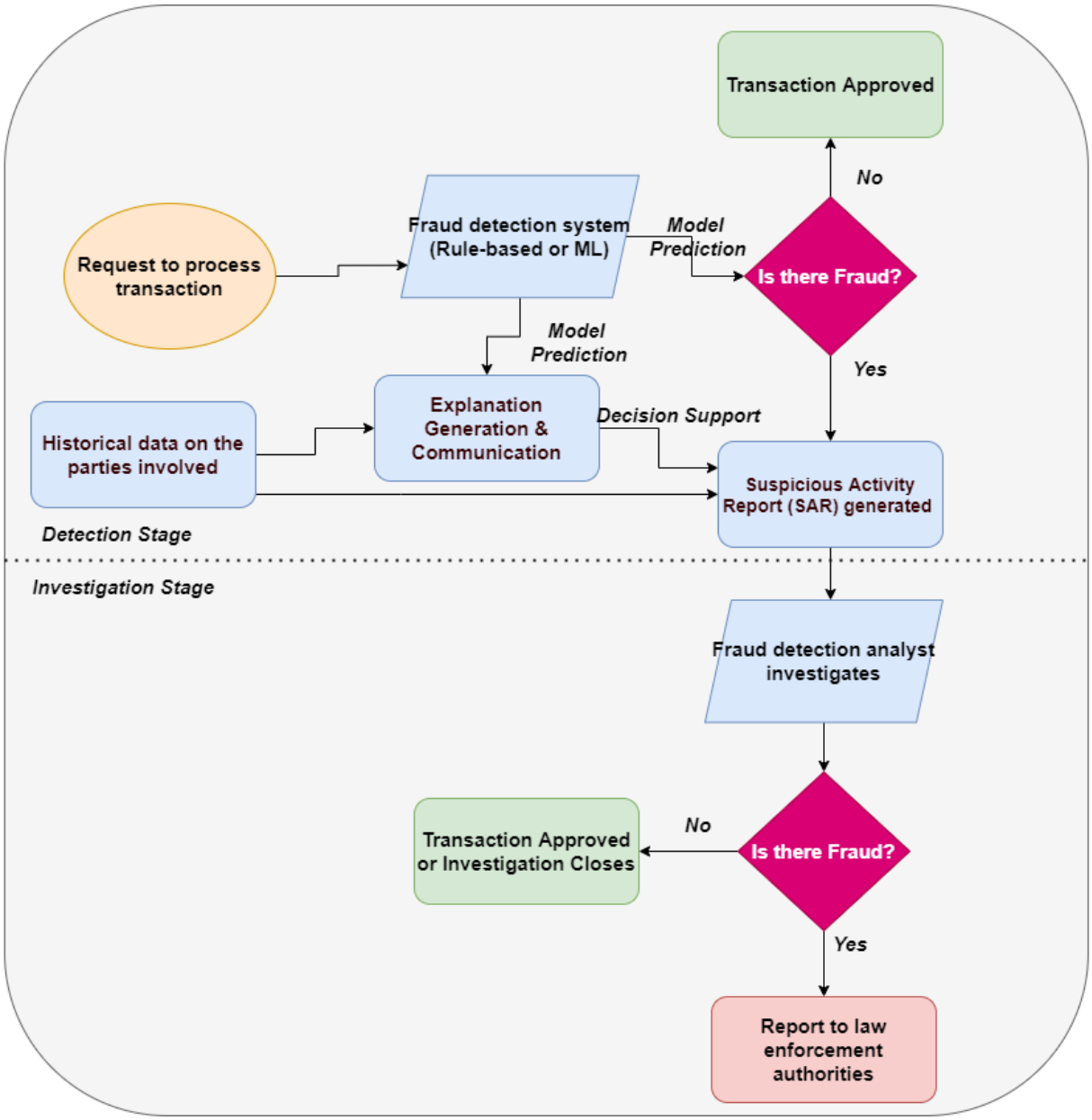


Figure 9.1: An Graphical Overview of Model Explanation Generation & Communication embedded within the transaction fraud monitoring process to augment the decision-making process of fraud detection analysts

A Python Code

The python code for all the experimentations conducted in this thesis research is stored in the GitHub Repository - https://github.com/codedevonfire/Python-Code---MSc-Thesis-EPA.git

Bibliography

(2016). REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

(2019). Guideline on the Anti-money Laundering and Anti-Terrorist Financing Act and the Sanctions Act. Technical report, De Netherlandse Bank.

(2019). Strategic Action Plan for Artificial Intelligence. Technical report, Ministry of Economic Affairs and Climate Policy.

(2021). Integrated gradients — TensorFlow Core.

(2021). Proposal for a regulation of the European parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts COM(2021) 206 final. Technical report, European Commission, Brussels.

Abdulla, N., R, R., and Mariam Varghese, S. (2015). International Journal of Scientific and Research Publications. *International Journal of Scientific and Research Publications*, 5(11).

Adadi, A. and Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160.

Ahmed, M., Ansar, K., Muckley, C. B., Khan, A., Anjum, A., and Talha, M. (2021). A semantic rule based digital fraud detection. *PeerJ Computer Science*, 7:e649.

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., and Aljaaf, A. J. (2020). A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. pages 3–21. Springer, Cham.

Anjomshoae, S., Najjar, A., Calvaresi, D., and Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review Robotics Track. *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, pages 1078–1088.

Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2019). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI.

Axelrod, R. M. (2017). Criminality and suspicious activity reports. *Journal of Financial Crime*, 24(3):461–471.

Barraclough, P. A., Hossain, M. A., Tahir, M. A., Sexton, G., and Aslam, N. (2013). Intelligent phishing detection and protection scheme for online transactions. *Expert Systems with Applications*, 40(11):4697–4706.

Barse, E. L., Kvarnström, H., and Jonsson, E. (2003). Synthesizing test data for fraud detection systems. *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 2003-January:384–394.

Bellovin, S. M., Dutta, P. K., and Reitinger, N. (2019). PRIVACY AND SYNTHETIC DATASETS. *22 Stan. Tech. L. Rev. 1*.

Bhatt, U., Andrus, M., Weller, A., and Xiang, A. (2020). Machine Learning Explainability for External Stakeholders. *arXiv*.

Bishop, C. M. (2006). Pattern recognition and machine learning - CERN Document Server.

Blanchart, P. (2021). An exact counterfactual-example-based approach to tree-ensemble models interpretability.

Byrd, J. and Lipton, Z. C. (2019). What is the Effect of Importance Weighting in Deep Learning? In *Proceedings of the 36th International Conference on Machine Learning, PMLR*, pages 872–881.

Cao, L. (2020). AI in Finance: A Review. *SSRN Electronic Journal*.

Cao, L., Yang, Q., and Yu, P. S. (2020). Data science and AI in FinTech: An overview.

Cartella, F., Anunciacao, O., Funabiki, Y., Yamaguchi, D., Akishita, T., and Elshocht, O. (2021). Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. *CEUR Workshop Proceedings*, 2808.

Casalicchio, G., Molnar, C., and Bischl, B. (2019). Visualizing the feature importance for black box models. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11051 LNAI, pages 655–670. Springer Verlag.

Chauhan, V. K., Dahiya, K., Sharma, A., and In, K. A. (2019). Problem formulations and solvers in linear SVM: a review. *Artificial Intelligence Review*, 52:803–855.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA. ACM.

Choo, K. K. R. (2015). Cryptocurrency and Virtual Currency: Corruption and Money Laundering/Terrorism Financing Risks? In *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*, pages 283–307. Elsevier Inc.

Collaris, D. and Van Wijk, J. J. (2020). ExplainExplore: Visual Exploration of Machine Learning Explanations. In *IEEE Pacific Visualization Symposium*, volume 2020-June, pages 26–35. IEEE Computer Society.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *ACM International Conference Proceeding Series*, volume 148, pages 233–240.

Deng, W., Huang, Z., Zhang, J., and Xu, J. (2021). A Data Mining Based System for Transaction Fraud Detection. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering, ICCECE 2021*, pages 542–545.

Doshi-Velez, F. and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning.

Dua, D. and Graff, C. (2017). UCI Machine Learning Repository: Adult Data Set.

Ehsan, U. and Riedl, M. O. (2020). Human-Centered Explainable AI: Towards a Reflective Sociotechnical Approach. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12424 LNCS, pages 449–466. Springer Science and Business Media Deutschland GmbH.

El Naqa, I. and Murphy, M. J. (2015). What Is Machine Learning? In *Machine Learning in Radiation Oncology*, pages 3–11. Springer International Publishing.

Fernandes, K., Cardoso, J. S., and Fernandes, J. (2017). Transfer learning with partial observability applied to cervical cancer screening. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10255 LNCS, pages 243–250. Springer Verlag.

Franklin, J. (2019). GDPR has kept AI ethical despite concerns. *International Financial Law Review*.

FSB (2017). Financial Stability Implications from FinTech: Supervisory and Regulatory Issues that Merit Authorities' Attention. Technical report, Financial Stability Board (FSB).

Gade, M., Williams, D., Mikkelsen, D., Richardson, B., Hasham, S., and Joshi, S. (2019). Transforming approaches to AML and financial crime. Technical report, McKinsey Company.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(4):463–484.

Gianey, H. K. and Choudhary, R. (2018). Comprehensive Review On Supervised Machine Learning Algorithms. In *Proceedings - 2017 International Conference on Machine Learning and Data Science, MLDS 2017*, volume 2018-January, pages 38–43. Institute of Electrical and Electronics Engineers Inc.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Pedreschi, D., and Giannotti, F. (2018). A Survey Of Methods For Explaining Black Box Models. *arXiv*.

Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G. Z. (2019). XAI-Explainable artificial intelligence. *Science Robotics*, 4(37).

Hall, M., Harborne, D., Tomsett, R., Galetic, V., Quintana-Amate, S., Nottle, A., and Preece, A. (2019). A Systematic Method to Understand Requirements for Explainable AI (XAI) Systems. *In Proceedings of the IJCAI Workshop on eXplainable Artificial Intelligence (XAI 2019)*.

Hashemi, M. and Fathi, A. (2020). PermuteAttack: Counterfactual Explanation of Machine Learning Credit Scorecards.

Hoffman, R. R., Mueller, S. T., Klein, G., and Litman, J. (2018). Metrics for Explainable AI: Challenges and Prospects. *arXiv*.

Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., and Wortman Vaughan, J. (2020). Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In *Conference on Human Factors in Computing Systems - Proceedings*. Association for Computing Machinery.

Klaise, J., Coca, A., Gorelli, M., Housley, A., Xie, V., Samiullah, C., and De La Iglesia Castra, D. (2020). SeldonIO/alibi: Algorithms for monitoring and explaining machine learning models.

Kong, J., Kowalczyk, W., Nguyen, D. A., Back, T., and Menzel, S. (2019). Hyperparameter Optimisation for Improving Classification under Class Imbalance. In *2019 IEEE Symposium Series on Computational Intelligence, SSCI 2019*, pages 3072–3078. Institute of Electrical and Electronics Engineers Inc.

Kou, Y., Lu, C. T., Sirwongwattana, S., and Huang, Y. P. (2004). Survey of fraud detection techniques. *Conference Proceeding - IEEE International Conference on Networking, Sensing and Control*, 2:749–754.

Langer, M., Oster, D., Kästner, L., Speith, T., Baum, K., Hermanns, H., Schmidt, E., and Sesing, A. (2021). What do we want from Explainable Artificial Intelligence (XAI)? A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artificial Intelligence*, page 103473.

Lawrence, N. D. (2019). Data Science and Digital Systems: The 3Ds of Machine Learning Systems Design.

Lentz, A., Siy, J. O., and Carraccio, C. (2021). AI-ssessment: Towards Assessment As a Sociotechnical System for Learning. *Academic medicine : journal of the Association of American Medical Colleges*, 96(7):S87–S88.

Li, H., Yu, L., and He, W. (2019). The Impact of GDPR on Global Technology Development. *https://doi.org/10.1080/1097198X.2019.1569186*, 22(1):1–6.

Li, L., Jamieson, K., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. Technical Report 185.

Liao, Q. V., Gruen, D., and Miller, S. (2020). Questioning the AI: Informing Design Practices for Explainable AI User Experiences. In *Conference on Human Factors in Computing Systems - Proceedings*. Association for Computing Machinery.

Lopez-Rojas, E., Elmir, A., and Axelsson, S. (2016). Paysim : A financial mobile money simulator for fraud detection. In Bruzzone, A., Jimenez, E., Louca, L., Zhang, L., and Longo, F., editors, *In: 28th European Modeling and Simulation Symposium, EMSS 2016*, pages 249–255, Dime. University of Genoa.

Lopez-Rojas, E. A. and Axelsson, S. (2012). Money Laundering Detection using Synthetic Data.

Lundberg, S. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 2017-December:4766–4775.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2019). Explainable AI for Trees: From Local Explanations to Global Understanding.

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., Liston, D. E., Low, D. K. W., Newman, S. F., Kim, J., and Lee, S. I. (2018). Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749–760.

Lv, L. T., Ji, N., and Zhang, J. L. (2008). A RBF neural network model for anti-money laundering. In *Proceedings of the 2008 International Conference on Wavelet Analysis and Pattern Recognition, ICWAPR*, volume 1, pages 209–215.

Mahajan, D., Tan, C., and Sharma, A. (2019). Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers.

Makki, S., Haque, R., Taher, Y., Assaghir, Z., Ditzler, G., Hacid, M. S., and Zeineddine, H. (2017). Fraud Analysis Approaches in the Age of Big Data-A Review of State of the Art. *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, pages 243–250.

Manju, N., Harish, B. S., and Prajwal, V. (2019). Ensemble feature selection and classification of internet traffic using xgboost classifier. *Computer Network and Information Security*, 7:37–44.

Mehlig, B. (2019). *Machine learning with neural networks - An introduction for scientists and engineers*.

Meng, C., Zhou, L., and Liu, B. (2020). A case study in credit fraud detection with SMOTE and XGboost. In *Journal of Physics: Conference Series*, volume 1601, page 52016. IOP Publishing Ltd.

Mintz, Y. and Brodie, R. (2019). Introduction to artificial intelligence in medicine. *https://doi.org/10.1080/13645706.2019.1575882*, 28(2):73–81.

Mishra, M. K. and Dash, R. (2014). A comparative study of chebyshev functional link artificial neural network, multi-layer perceptron and decision tree for credit card fraud detection. In *Proceedings - 2014 13th International Conference on Information Technology, ICIT 2014*, pages 228–233. Institute of Electrical and Electronics Engineers Inc.

Mittelstadt, B., Russell, C., and Wachter, S. (2019). Explaining explanations in AI. In *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, pages 279–288. Association for Computing Machinery, Inc.

Modi, K. and Dayma, R. (2018). Review on fraud detection methods in credit card transactions. *Proceedings of 2017 International Conference on Intelligent Computing and Control, I2C2 2017*, 2018-January:1–5.

Molnar, C. (2020). *Interpretable Machine Learning*. lulu.com.

More, A. (2016). Survey of resampling techniques for improving classification performance in unbalanced datasets.

Mothilal, R. K., Mahajan, D., Tan, C., and Sharma, A. (2020a). Towards Unifying Feature Attribution and Counterfactual Explanations: Different Means to the Same End.

Mothilal, R. K., Sharma, A., and Tan, C. (2020b). Explaining machine learning classifiers through diverse counterfactual explanations. In *FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617. Association for Computing Machinery, Inc.

Mubalaike, A. and Adali, E. (2017). Multilayer perceptron neural network technique for fraud detection. In *2nd International Conference on Computer Science and Engineering, UBMK 2017*, pages 383–387. Institute of Electrical and Electronics Engineers Inc.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*, 116(44):22071–22080.

Naheem, M. A. (2018). TBML suspicious activity reports – a financial intelligence unit perspective. *Journal of Financial Crime*, 25(3):721–733.

Nami, S. and Shajari, M. (2018). Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors. *Expert Systems with Applications*, 110:381–392.

Oza, A. (2018). Fraud detection using machine learning. Technical report, Standford University.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian Optimization Algorithm. *In Proceedings of the genetic and evolutionary computation conference GECCO-99*, 1:525–532.

Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A Comprehensive Survey of Data Mining-based Fraud Detection Research. *Computers in Human Behavior*, 28(3):1002–1013.

Ponomareva, N., Radpour, S., Hendry, G., Haykal, S., Colthurst, T., Mitrichev, P., and Grushetsky, A. (2017). TF Boosted Trees: A scalable TensorFlow based framework for gradient boosting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10536 LNAI:423–427.

Putatunda, S. and Rama, K. (2018). A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost. *dl.acm.org*, pages 6–10.

Ramchoun, H., Amine, M., and Idrissi, J. (2016). Multilayer Perceptron: Architecture Optimization and Training multi-criteria learning and nonlinear optimization View project. *Article in International Journal of Interactive Multimedia and Artificial Intelligence*, 4:1–26.

Ranjan, G. S., Kumar Verma, A., and Radhika, S. (2019). K-Nearest Neighbors and Grid Search CV Based Real Time Fault Monitoring System for Industries. In *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019*. Institute of Electrical and Electronics Engineers Inc.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 1135–1144. Association for Computing Machinery.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence 2019 1:5*, 1(5):206–215.

Russell, C. (2019). Efficient search for diverse coherent explanations. In *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, pages 20–28. Association for Computing Machinery, Inc.

Safdari, A., Nurani, M. S., and Aghajani, K. (2015). Social Impact of money laundering. *Asian Journal of Research in Social Sciences and Humanities*, 5(1).

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

Sankaran, R. and Chakraborty, S. (2020). Why customers make mobile payments? Applying a means-end chain approach. *Marketing Intelligence & Planning*, 39(1):109–124.

Sanneman, L. and Shah, J. A. (2020). A Situation Awareness-Based Framework for Design and Evaluation of Explainable AI. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12175 LNAI, pages 94–110. Springer.

Schleich, M., Geng, Z., Zhang, Y., and Suciu, D. (2021). GeCo: Quality Counterfactual Explanations in Real Time.

Shapley, L. S. (2016). 17. A Value for n-Person Games. In *Contributions to the Theory of Games (AM-28), Volume II*, pages 307–318. Princeton University Press.

Siebert, J., Joeckel, L., Heidrich, J., Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R., and Aoyama, M. (2020). Towards Guidelines for Assessing Qualities of Machine Learning Systems. *Communications in Computer and Information Science*, 1266 CCIS:17–31.

Štrumbelj, E. and Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic Attribution for Deep Networks. *34th International Conference on Machine Learning, ICML 2017*, 7:5109–5118.

Suresh, S., Sundararajan, N., and Savitha, R. (2013). Supervised Learning with Complex-valued Neural Networks. *Springer*.

Syahadiyanti, L. and Subriadi, A. P. (2018). International Journal of Economics and Financial Issues Diffusion of Innovation Theory Utilization Online Financial Transaction: Literature Review. *International Journal of Economics and Financial Issues*, 8(3):219–226.

Toby Sterling (2021). ABN Amro money laundering probe close to settlement, FD reports — Reuters.

Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S. (2018). Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects.

Van der Burgt, J. (2019). General principles for the use of Artificial Intelligence in the financial sector. Technical report, De Nederlandsche Bank.

Van Looveren, A. and Klaise, J. (2019). Interpretable Counterfactual Explanations Guided by Prototypes.

Wachter, S., Mittelstadt, B., and Floridi, L. (2017). Transparent, explainable, and accountable AI for robotics. *Science Robotics*, 2(6):31.

Wachter, S., Mittelstadt, B., and Russell, C. (2018). Automated Decisions and the GDPR' (2018) 31(2) Harvard Journal of Law & Technology (Harvard JOLT) 841. MLA 8th ed. Wachter, Sandra, et al. Automated Decisions and the GDPR. Technical Report 2.

Watkins, R. C., Reynolds, K. M., Demara, R., Georgiopoulos, M., Gonzalez, A., and Eaglin, R. (2003). Tracking dirty proceeds: Exploring data mining technologies as tools to investigate money laundering. *Police Practice and Research*, 4(2):163–178.

Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C. E., and Schardl, T. B. (2018). Scalable Graph Learning for Anti-Money Laundering: A First Look.

Wolf, C. T. (2019). Explainability scenarios: Towards scenario-based XAI design. In *International Conference on Intelligent User Interfaces, Proceedings IUI*, volume Part F1476, pages 252–257. Association for Computing Machinery.

Worobec, K. (2020). FRAUD-THE FACTS 2020 The definitive overview of payment industry fraud. Technical report, UK Finance.

Wrenninge, M. and Unger, J. (2018). Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing.

Wu, Y., Dobriban, E., and Davidson, S. (2020). DeltaGrad: Rapid retraining of machine learning models.

Xie, N., Ras, G., van Gerven, M., and Doran, D. (2020). Explainable Deep Learning: A Field Guide for the Uninitiated.

Zareapoor, M. and Yang, J. (2017). A Novel Strategy for Mining Highly Imbalanced Data in Credit Card Transactions. *Changed publisher: TSI Press*, pages 1–7.

Zhang, X. P. S. and Kedmey, D. (2018). A budding romance: Finance and AI. *IEEE Multimedia*, 25(4):79–83.

Zhang, Y. and Trubey, P. (2019). Machine Learning and Sampling Scheme: An Empirical Study of Money Laundering Detection. *Computational Economics*, 54(3):1043–1063.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.