NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

A NOVEL METHODOLOGY FOR EVALUATING MACHINE LEARNING PIPELINES APPLIED TO THE CONTEXT OF FINANCIAL FRAUD DETECTION

Samuel Santos

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

A NOVEL METHODOLOGY FOR EVALUATING MACHINE LEARNING PIPELINES APPLIED TO THE CONTEXT OF FINANCIAL FRAUD DETECTION

by

Samuel Santos

Master Thesis presented as a partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science.

Supervised by

Leonardo Vanneschi, PhD, NOVA Information Management School

Nuno Rosa, PhD, NOVA Information Management School

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 15-07-2024

DEDICATION

To my family and friends. For the love, belief, and support that I received since forever. My heart and soul.

To the inspiring, passionate, and brilliant professors I had the luck to meet in my life. For making me want to strive for excellence. My vision and mind.

To the colleagues with whom I shared this journey. For your unfailing trust, companionship, support and friendship. My feet, my hands. My wings.

ACKNOWLEDGEMENTS

First and foremost, I thank my supervisors, Professor Leonardo Vanneschi and Professor Nuno Rosa, for their significant contributions from the very start and for making the time during this long process to help, challenge, encourage, and always demand the best while showing support and enthusiasm.

Several colleagues made a significant contribution by having the unbelievable patience to listen to some of my ideas and doubts, then ask questions, disagree, and argue. First and foremost, Adriana Monteiro, Marta Dinis, Inês Magessi, and Isabel Dias. Also Luís Fernandes, Iryna Savchuk, Patrícia Morais, Rita Soares, Felix Gaber, Janaina Santos, Inês Rocha, Alex Santander, Jorge Ceja, Jaime Abreu. Thank you for your help on the thesis and for being with me on this journey. Professor Ricardo Santos and Professor Bruno Damásio generously took the time to clarify some specific but relevant questions. Elvira Costa and Telmo Fonte, I appreciate your unfailing support and enthusiasm; NOVA IMS has the best library service in the world.

I was told from the beginning, as were all my colleagues, that the supervisors would help, guide, and contribute, but in the end, it would be my work, my concern and my responsibility. My thesis. This was not at all my experience. So, while what is written in the thesis is solely my responsibility, it does feel that I'm somehow taking credit for what was a large group project.

ABSTRACT

Financial fraud is a crime that extracts value from society and creates disadvantages for economic agents operating in good faith. As the means used for payment evolved, so did fraud and fraud detection. Earlier software in this area would employ some fixed conditions that were interpreted as suspicious. When a certain form of fraud is successfully detected and prevented, perpetrators will look for new ways to deceive the system, and the cycle repeat itself. Artificial Intelligence moves away from rules that need to be encoded in applications and uses data to find the patterns that denounce the presence of fraud. Extensive research is being conducted in all areas that relate to fraud. It is not common, however, for a study to present a broader technical and quantitative analysis of the way in which these techniques combine and interact. In this thesis, a thorough analysis is conducted on several steps along the Machine Learning processing pipeline, starting with different datasets and including key functions such as scaling, null value imputation, resampling, feature engineering, feature selection, ending with the supervised model. These results are compiled and analyzed by a multilinear regression that allows for the measurement of the impact of individual options on a complex pipeline. In the literature, this approach has not been used for supervised learning, namely in the context of fraud detection. This study demonstrates the importance of some aspects, such as the quality of the data used, the relevance of feature engineering, and the fact that even if a certain step does not make a significant positive contribution to results, poor choices can still significantly hinder results. This thesis will eventually be beneficial for researchers or other types of professionals trying to better understand the ways in which supervised learning can help in fraud detection. From a business perspective, the structure used in this study can be replicated and help increase the automation of the critical process of model retraining and can help in directing resources and investment for the steps that yield the best returns. The methodology proposed is flexible, allowing for some parts to be removed and for new parts to be added. It can be used in binary or multiclass classification, as well as in regression problems.

KEYWORDS

Fraud Detection; Compare ML techniques; Imbalanced Dataset; Multilinear regression

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ADASYN	Adaptive Synthetic Sampling
AI	Artificial Intelligence
AIS	Artificial Immune Systems
AL	Active Learning
ANN	Artificial Neural Networks
AUROC	Area Under the Receiver Operating Characteristic (curve)
AutoML	Automated Machine Learning
BLUE	Best Linear Unbiased Estimator
CC	Credit Card
CNP	Card Not Present (transactions)
CRISP-DM	Cross-Industry Standard Process for Data Mining
DM	Data Mining
DOF	Degree(s) Of Freedom
EFB	Exclusive Feature Bundling
FDS	Fraud Detection System
FN	False Negatives
FP	False Positives
GBDT	Gradient-Boosting Decision Trees
GDP	Gross Domestic Product
GOSS	Gradient-based One-Sided Sampling
HMM	Hidden Markov Model
HPT	Hyper Parameter Tuning
IG	Information Gain
IQR	Inter-Quartile Range
IT	Information Technology

LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
LightGBM	Light Gradient Boosting Machine
LR	Linear Regression
ML	Machine Learning
MLR	Multi Linear Regression
PCA	Principal Component Analysis
RFE	Recursive Feature Elimination
RFM	Recency-Frequency-Monetary value
SLR	Simple Linear Regression
SMOTE	Synthetic Monitory Oversampling Technique
SVM	Support Vector Machines
TN	True Negatives
TP	True Positives
XGBoost	Extreme Gradient Boosting

1 INTRODUCTION

1.1 EVOLUTION OF MONEY AND PAYMENTS

Payment is the transfer of value from one agent to another (*The New Palgrave Dictionary of Economics*, 2008, p. 337).

In the early days of commerce, payments were made through bartering (Boel, 2019). This required that both parties involved would be interested specifically in what the other had to offer and that the valuation would be similar. Otherwise, only one of the agents would be interested in the exchange. Assets owned had therefore reduced liquidity, as they could not be converted at short notice (Hicks, 1962, p. 791). Furthermore, most of these assets would not be a convenient or secure way to store value for later use.

It is argued (Caton & Harwick, 2022) that the evolution of money is driven by the need to reduce transaction costs, and bartering had substantial transaction costs. This direct exchange of assets gave way to payments that involved commodities such as salt or iron (Heron, 2017) that would be easier to transact and had advantages such as being durable, easily divisible, and having intrinsic value. In ancient Rome, salt was used for paying soldiers and civil servants (Monsen, 1993). The word salary that we still use today comes from the Latin *salarium*, a word derived from *sal* (Latin for salt). The use of salt and other alternatives removed the need for direct exchange, with the added benefit that these resources could be used as a means to store value, allowing for the consumption to be postponed.

Money was invented to provide liquidity, facilitate transactions, and store value reliably. In fact, liquidity is used to denote the property of "being like cash" (Makower & Marschak, 1938, p. 284). Metal coins started being produced 600 years BC (Banco de Portugal, 2020) and became popular in the Roman Empire. They were manufactured manually until the Industrial Revolution.

Despite presenting significant advantages, large amounts of metal coins were necessarily heavy and, therefore, not practical to carry.

Metals such as copper, which were used to produce metal coins, were becoming scarce in China in the 9th and 10th centuries and were needed for the war effort involving several kingdoms of the region. Besides the problem that they were hard to obtain, there was also the realization that enemies, upon conquering a certain territory, could melt the coins of the captured region, using the metal to produce more weapons. This pressure led to the creation of several paper-based alternatives (Horesh, 2012), which would evolve into the paper bills that were invented in the 11th century. In the Western world, Sweden introduced notes (*kreditivsedlar*) in 1661. By 1860, banks had mechanized the production of notes, eliminating the need for an individual, manual intervention.

With the increased digitalization of society, payments and other financial transactions are nowadays mostly digital – credit card payments, online transactions, money transfers, mobile payments, and cryptocurrency.

1.2 FRAUD AND FRAUD DETECTION

Fraud is "an uncommon, well-considered, time-evolving, carefully organized and imperceptibly concealed crime" (Van Vlasselaer et al., 2017). The underlying objective behind this crime is the unlawful capture of value belonging to one entity by another entity without the consent or even knowledge of the first.

Fraud attempts to capture part of the value in physical and digital money. As soon as money was used as a means to store value, there was an incentive to obtain it inappropriately, and there was a need to prevent that from happening. Technological progress in money and payments is naturally accompanied by an evolution of fraud mechanisms and their prevention counterparts.

Before the digitization of money, in 1925, Alves dos Reis managed to forge documents and obtain a shipment of notes produced by the English security printing firm employed by Portugal's central Bank, after persuading this firm that he was mandated by Portugal's Central Bank. This fraud was of an unprecedented dimension, estimated to be 0.88% of the country's gross domestic product (GDP) at that time (Wigan, 2004). There is even speculation that the introduction of such a large account of bills in circulation raised inflation even further (it was already high at the time) and contributed to accelerating the economic and political unrest that led to the change of political regime in 1926.

Modern bills have many characteristics that try to make them difficult and costly to imitate, including some that are kept secret.

As money becomes digitized, fraudsters need to operate in this new environment and adapt. The entities trying to detect fraud need to investigate the digital footprint that is recorded on information technology (IT) systems, in order to improve the predictive capabilities of the systems used to detect fraud.

If a fraud has already occurred, it might be costly or even impossible to revert the operation and recover the value lost in the fraud. It is therefore of paramount importance to detect fraud while it is being attempted, so that the entities affected may prevent it from happening, rather than searching for a reparation afterward.

1.3 RESEARCH ON FRAUD DETECTION

There has been much progress in automated fraud detection, from the earlier statistical methods to more sophisticated computational controls implemented in IT systems. It comes as no surprise that artificial intelligence (AI) is being used to detect and prevent fraud. In fact, given the significant economic loss that is involved, naturally, the effort and investment in this

area by companies and banks is very relevant, and abundant research is being conducted on the topic.

CRISP-DM, the *de facto standard* in Machine Learning projects, defines six phases: Business Understanding, Data understanding, Data preparation, Modelling, Evaluation and Deployment. This thesis focuses on the phases that directly involve handling data, from its collection to predictions on previously unseen data. The handling of data in supervised learning, including fraud detection, requires several steps to be taken. These start with data collection, then data preprocessing (involving several sequential layers of processing), and finally, the training of a supervised model. This sequence can be seen as a pipeline in which data enters, then is pushed forward and transformed, and in the end, it is used to train the model so that it learns and becomes capable of making better predictions (train data) or to make predictions on new data (validation/test data). Better predictions depend on the model, on the quality of the data, and on the transformations that are operated inside the pipeline between these two points.

As the pipeline is fine-tuned, it becomes increasingly better at predicting outcomes. On the other hand, if one of the processing steps is faulty or unoptimized, it can compromise the pipeline's capacity to make good predictions. Yet, most studies scrutinize in detail just one part of the pipeline and consider the rest to be fixed and independent from what is under analysis in the specific study.

Some studies present innovative techniques and measure the results without comparing them to a benchmark result (Soltani Halvaiee & Akbari, 2014; Jovanovic et al., 2022). Other studies consider two or more alternatives and evaluate them (Thimonier et al., 2023; Rtayli & Enneya, 2020), but still, these alternatives are focused on a specific part of the Machine Learning pipeline and consider all other steps to be fixed.

In some cases, a comparison is made by simultaneously defining several alternatives for a certain step, with most studies concentrating on the imbalanced dataset issue (Makki et al., 2019) or on experimentation with different models (Ranjon Das et al., 2023). Some other studies analyze options for a certain step and then measure results using more than one model to circumvent biases that could be model-dependent. Again, all other parts of the pipeline are fixed.

Some recent approaches in the realm of AutoML (Urbanowicz et al., 2023) implement and evaluate a fully supervised machine-learning pipeline and compare different configurations.

In all of those cases, it is the overall performance of the system that is being assessed. It is plausible that a certain model obtains worse results under a certain fixed option, but would have a better performance if the fixed option had been a different one. This is typically unaccounted for in the results of such studies. When a complete pipeline is being evaluated, there is no assessment of the contribution of each individual option to the overall results.

1.4 RESEARCH GAP AND MOTIVATION

The approaches mentioned have the advantage that by significantly reducing the number of possible configurations, it becomes possible to experiment exhaustively with the few that remain. The pipelines created for these purposes are exactly equal, except for the specific part that is being studied.

This approach corresponds to establishing a "ceteris paribus" assumption. This is common in academic fields such as economics and statistics; the concept is that by changing the value of one variable, we determine its effect on a target variable, and the key assumption is that the remaining variables will not change. This assumption of variable independence in practice is naïve, even unrealistic, as it is well established that changes in different parts of the pipeline will have positive or negative interactions, improving or degrading the results obtained. This impact is not captured by studies conducted in this fashion.

This research gap inspired the idea of conducting a study in two distinct stages.

The first is the Machine Learning (ML) pipeline used for fraud detection, a binary classification problem in which it is necessary to identify whether a transaction is fraudulent (positive label) or a legitimate transaction (negative label). For this stage, a pipeline is built, featuring 9 different steps. Each step is called a Degree of Freedom (DOF) of the pipeline, as it is at this level that distinct paths (options) are presented, and there is a decision to follow one of them. E.g., for the DOF corresponding to scaling data, four options will be available, and each pipeline will incorporate one of them: ignore scaling, use standard, robust or MinMax scaler. The pipeline will be built with the combination of the 9 choices, each corresponding to one DOF. This pipeline will be used on the binary classification problem, its results will be assessed using a metric and recorded, along with the 9 choices that define the pipeline's configuration.

In this first stage the options available at each DOF will allow for the construction of more than 10.000 pipelines in which no combination of all DOF will be the same, making these pipelines unique.

The second stage will be a regression problem. The target will be the continuous variable corresponding to the metric recorded in the first stage for each pipeline, and the options in the DOF of the pipeline will constitute the independent variables used to explain that metric. The objective is to explain the contribution of each individual option to the overall result.

By using this two-stage approach, it is expected that insights into what works best and what doesn't contribute to results will be determined objectively. Moreover, this approach can be extended to easily evaluate the impact of using different techniques without having to impose "ceteris paribus" conditions on the experimentation. E.g., a solid assessment of the impact of using geometric SMOTE for sampling can be obtained by implementing it and making several runs with different configurations, yet without the need to experiment all configurations.

1.5 EXPECTED RESULTS

Different configurations will produce different results during the first stage's runs. The most relevant expectation in this stage is that there will be some variance in the results, so it will be possible to study the causes of that variance.

The most relevant expectations relate to the second phase, as each option in the pipeline is being evaluated.

It is expected that some options will yield better results than others, and so the pipelines in which they are present will obtain better scores. This may not be easily identified during the first stage, as many options are being changed at the same time, and some may be lowering the results. Through the second phase, it is expected that these results will become visible. It can be anticipated that some options will have a statistically significant impact, either positive or negative, while some others will show results that are not statistically significant. By keeping only the options whose results are both statistically significant and positive, an equivalent pipeline can be built, but using a streamlined configuration.

As the system is a combination of all parts, understanding the individual elements may provide insights into the overall system in a way that hopefully will be significant, meaningful, actionable, and explainable.

The supervised model chosen for phase two, to study the regression, is the multilinear regression model (MLR) due to two relevant characteristics: it provides high explainability by associating a weight (a Beta) to each independent variable, with this weight measuring the impact on the target variable, and the second characteristic is that it associates a p-value to that weight, informing the degree of statistical significance of that measure.

An MLR is a statistical tool that takes as inputs observations with several dimensions (the explanatory variables) and determines the contribution of each dimension to the metric that is being predicted (the dependent variable). Contrary to measuring the impact of changing just one variable and assuming no interactions, a multiple linear regression adequately considers the interactions between explanatory variables and allows for measuring the individual impact of each variable, therefore providing results that are more robust.

The MLR creates a “baseline result” which is the prediction for the dependent variable when all explanatory variables are 0. As the variables employed for defining the options are categorical in nature, one-hot encoding must be performed. This technique converts a categorical variable with n possible values into a set of n-1 binary variables. Only one of these variables can be True for any observation. The category that was removed will be represented by all variables being False. This is the category that will be present in the baseline.

The result calculated by the MLR for each variable expresses how much using that option for a certain DOF, instead of the baseline option for that DOF, would benefit or hinder

performance. From theory, we know that the model obtained by the MLR will be the best linear unbiased estimator (BLUE).

This approach to measuring the performance of different options on an ML pipeline for fraud detection is novel, albeit based on a well-established tool, the multilinear regression, and applied on a well-known problem, fraud detection.

As the variable being measured only takes values in the interval [0,1], it is expected that only small weights will be associated with each option.

The datasets employed are highly unbalanced, and the metric chosen is F1 score-weighted. Therefore, it is expected that the baseline will have a relatively high score, implying that the alternatives at each DOF will have small effects.

2 LITERATURE REVIEW

Building on existing research, for the purpose of this thesis an extensive and broad spectrum literature review was conducted on Fraud in the context of Artificial Intelligence (AI).

2.1 PAYMENTS FRAUD

Fraud has distinct definitions (Akers & Gissel, 2006) covering the legal aspect, the intent to misrepresent and deceive, and the misappropriation of valuable assets.

Fraud can be defined as the "*intentional perversion of truth in order to induce another to part with something of value or to surrender a legal right*" ('Merriam-Webster Dictionary, Definition of Fraud', 2024) or as "*the crime of cheating somebody in order to get money or goods illegally*" (Oxford Advanced Learner's Dictionary, Fraud - Definition, 2024) or as "*any intentional act or omission designed to deceive others, resulting in the victim suffering a loss and/or the perpetrator achieving a gain*" (Cotton et al., 2023).

The cost of fraud is economic (reduced operational effectiveness), legal (deprivation of resources from rightful claimants) and psychological (damaged morale and reduced confidence) (Alexopoulos et al., 2007).

Several typologies of fraud exist (Akers & Gissel, 2006; Baesens, 2022); this thesis will focus on Payment fraud, defined as the unlawful act of forging, stealing or manipulating personal or financial information to make payments on behalf of someone else, or to prevent real monetary transactions between the recipient of a good or service and the provider (Bockel-Rickermann et al., 2023). Only payments done through electronic means and involving a financial services entity will be considered.

Fraud detection in this thesis will be based on "fraud analytics", defined (Bockel-Rickermann et al., 2023) as the use of data-driven methods to discover, recognize and detect fraudulent activities in sets or streams of data.

Payment fraud includes credit and debit card fraud, as well as online payments. Of these, credit card fraud is the largest part, representing an estimated loss of 32.4 thousand million USD in 2021, up from 9.84 thousand million USD in 2011, having increased 3.3 times in 10 years, and making it the fastest-growing form of identity theft. It is estimated to reach 43 thousand million USD by 2026 (Bolton & Hand, 2002).

The impact of payment fraud extends beyond the financial services industry and significantly affects governments, merchants, and other corporations, as well as individual financial services users. With transactions and even currencies becoming increasingly digital, manual detection of fraud is not only time-consuming, expensive, and inaccurate, but it also becomes impractical.

Out of ten different credit card (CC) fraud types identified (Jain et al., 2019), Card-not-present (CNP) transactions (including e-commerce transactions, phone transactions, and manually entered card numbers) are the least secure and account for 65% of all CC fraud losses. 46% of all credit card fraud losses occur in the USA. The UK, France, and Ireland are the European countries with the highest rates of CC fraud (Bolton & Hand, 2002).

2.2 FRAUD DETECTION SYSTEMS (FDS)

Financial institutions started by employing auditing and rule-based methods, then evolved to statistical and computational methods, and moved into Artificial Intelligence (AI), using Machine Learning (ML) and Data Mining (DM) models (West & Bhattacharya, 2016). While the application of AI is nowadays a *de facto* standard, the application of AI/Expert Systems for real-time credit card fraud deterrence, replacing end-of-the-day reports and offline proceedings, can be found, for instance, in a Canadian bank, as early as the beginning of the '90s (Leonard, 1993).

Models extensively analyzed in academic papers and in the industry include Decision Trees, Neural Networks, Support Vector Machines, K Nearest Neighbors, Logistic Regression, Naïve Bayes, and Random Forest (Baesens, Höppner, & Verdonck, 2021; Maher, 2020). Besides these supervised learning models, there is a relevant contribution to the end result given by unsupervised models, especially in engineering features that prove useful for the supervised models. These will be mentioned below, as well as the semi-supervised models that aim to combine the advantages of both supervised and unsupervised models.

Some Fraud Detection Systems are also created from models that are nature-inspired.

2.3 FRAUD DETECTION CHALLENGES

One significant challenge in fraud detection is the need for predictions to occur in real time or near real time (H. Wang et al., 2022).

Several other key challenges are identified in the literature for financial fraud (Bockel-Rickermann et al., 2023) and for credit card fraud (Patel, 2023). These include Class imbalance, concept drift, scalability, presence of unlabeled data, and limited data availability.

2.3.1 Need for real-time processing

A Fraud Detection System (FDS) that is able to identify a transaction as fraudulent, while the transaction is being processed, will be actually capable of deterring the fraud. If, however, the FDS is analyzing transactions that were already authorized, it will be uncovering frauds that already occurred and whose loss can now be acknowledged, but not prevented.

Real-time detection can be performed by several participants in the transaction, at different moments,as illustrated in the processes in Figure 2.1 and Figure 2.2 (Y. Wang et al., 2016).

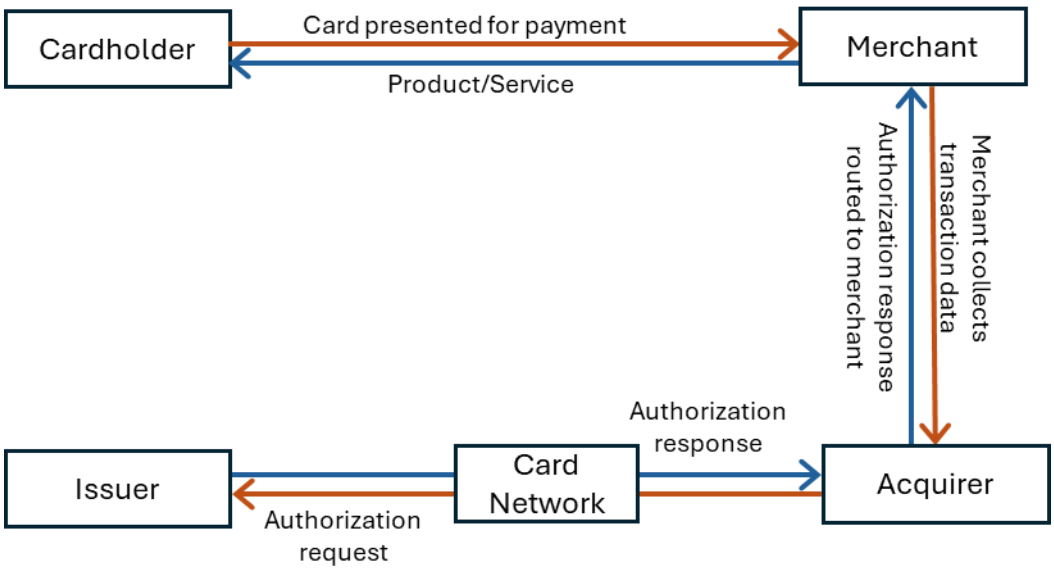

Figure 2.1 - Debit/Credit card Payment processing (Y. Wang et al., 2016)

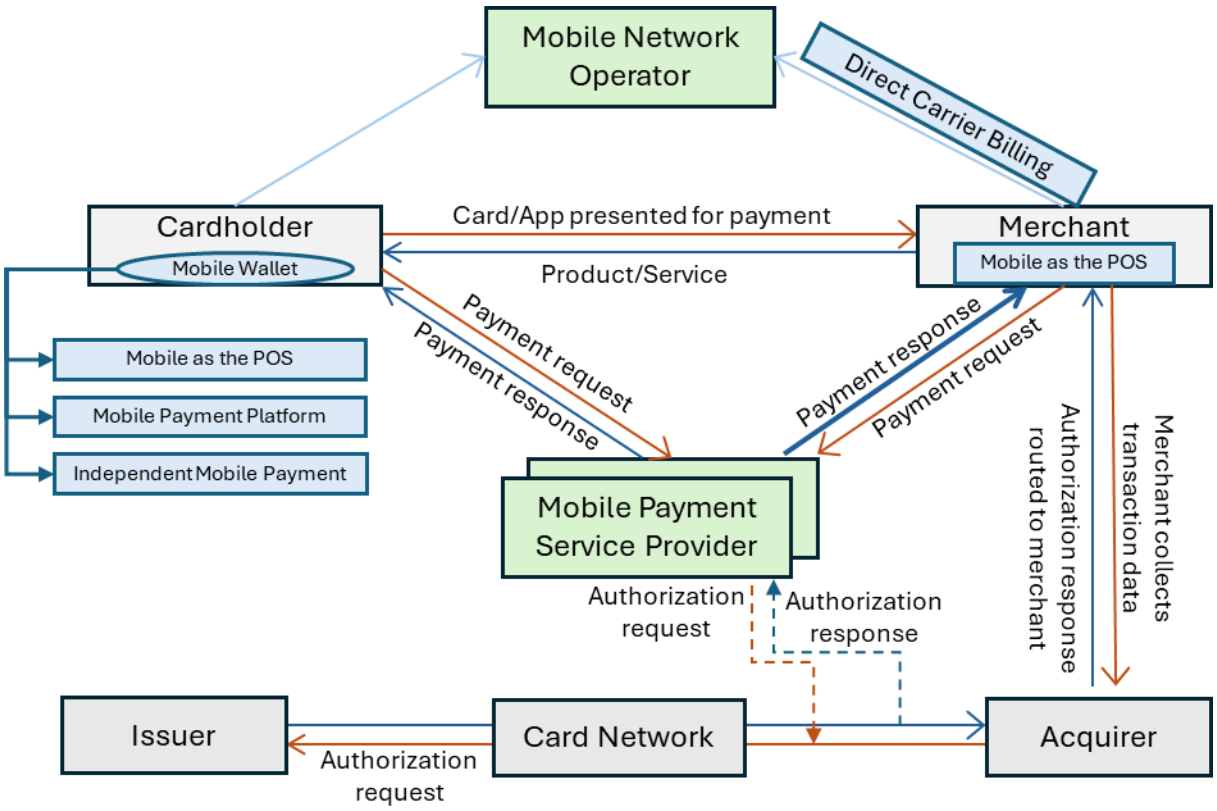

Figure 2.2 - Mobile Payment processing (Y. Wang et al., 2016)

2.3.2 Class Imbalance

Class imbalance occurs when a class (or classes) has a significantly higher number of occurrences than another class (or classes). There is a vast corpus of work done on class imbalance, as it is, in fact, a common condition in classification problems.

Class imbalance poses two challenges. The first is the risk of overfitting; some models will tend to overfit when faced with imbalanced datasets, as is the case with Decision Trees (Khare & Sait, 2018, p. 832) and with Logistic Regression (A. Aslam & Hussain, 2024, p. 3). Moreover, some of the techniques employed to compensate for class imbalance could further promote overfitting, as is the case of unsophisticated oversampling techniques, such as random oversampling (Su et al., 2024). The second challenge is that, when faced with a class label with very few instances, the model might not have enough information to learn the concept of the minority class accurately (Bauder et al., 2018).

One technique to optimize results for problems with class imbalance is called cost-sensitive learning; it defines a cost-matrix that penalizes differently each type of error. This implementation will be model-dependent; for instance, for support vector machines (SVM) the cost parameter c can be defined to be inversely proportional to the number of samples in each class (Anand et al., 2010). The cost-sensitive approach has been used specifically for credit card fraud detection, in this instance using a weighted Random Forest (Devi et al., 2019).

Other common techniques include undersampling (B. Liu & Tsoumakas, 2020), in which elements of the majority class are discarded until the imbalance is reduced. Oversampling is the symmetric technique; it involves creating new elements in the minority class. There are numerous oversampling variations (Barua et al., 2011; Fan et al., 2011; Ai et al., 2015; Douzas et al., 2018). The risk of overfitting mentioned before is relevant for implementations of oversampling that take a naïve approach by simply reproducing existing data points. Some other techniques are more sophisticated, as is the case of Synthetic Monitory Oversampling Technique or SMOTE (Chawla et al., 2002) and its variations, including K-means SMOTE (Fonseca et al., 2021), and Adaptive Synthetic Sampling or ADASYN (He et al., 2008), that takes into account the learning difficulty for each minority class.

2.3.3 Concept Drift

Concept drift may occur when the distribution of a class in a dataset is time-dependent (Gama et al., 2014). As forms of payment fraud are detected and deterred, fraudsters will adapt and experiment with different approaches to perpetrate fraud (Leite, Miguel Lobo Pinto, 2020, p.2), causing concept drift. If the FDS fails to adapt, it may become very good at preventing frauds that are no longer actively explored precisely because they are unsuccessful, but at the same time, fail to identify new forms of fraud. This degradation will result in higher numbers of false negatives (Dal Pozzolo et al., 2015). Concept drift has implications in the way that data training is approached.

2.3.4 Scalability

Increased digitization and wider adoption of digital payments results in more agents sending data for validation, while at the same time more data is being sent for each transaction. Scalability is therefore a key requirement for any model to be used in a real-world production system (Melo-Acosta et al., 2017). Scalability in this context means not only being able to train and validate models using more data and process more transactions but also getting results from the models in a time lapse that is adequate to the business needs.

2.3.5 Unlabeled or partially unlabeled data

A dataset is considered partially unlabeled when only a subset has labels or when for some labeled data points, we are not sure if the label is correct. For fraud detection, we have to deal with the reality of label scarcity (Leite, Miguel Lobo Pinto, 2020). In datasets used for supervised learning, there is usually a human expert manually assigning labels to data. Doing this for electronic payments is evidently unfeasible due to the sheer volume of transactions that are processed in a small amount of time. This means that only a small subset of transactions will be annotated, so that it can be used as train data.

Active Learning is a field of Machine Learning that deals with this reality, trying to obtain robust predictions from scarce labeled observations when labeling all observations is too expensive, time-consuming, or otherwise impractical. This is common in scenarios where the positive class is very rare and/or when data must be manually labeled, and that would represent a very significant cost. In these scenarios, we have a dataset with some labeled records and a large number of unlabeled records. A semi-supervised approach can be employed, in order to take advantage of both the labeled and unlabeled data. One approach (Hu et al., 2019) is to use unsupervised representative learning on the unlabeled data and supervised learning on the labeled data.

2.3.6 Real-world data unavailability

Besides data label scarcity, data availability for payment fraud is also an issue due to concerns about confidentiality and privacy (Saia & Carta, 2019). This significantly restricts access and permission to use real data. This is quite apparent in the fact that recent real datasets are hard to come by. This causes researchers to resort to synthetic or to outdated data, which evidently is not ideal.

2.4 FEATURE SELECTION AND FEATURE ENGINEERING FOR FRAUD DETECTION

Feature engineering influences the effectiveness of the models employed for fraud detection (Abedin et al., 2023); feature selection focuses on the efficiency of the system. Feature engineering is the creation of synthetic features by transforming the raw data, while feature selection is the determination of what features have a positive/significant contribution to the

model's performance (Correa Bahnsen et al., 2016; Long et al., 2019; Abedin et al., 2021; H. Zhang et al., 2021; X. Zhang et al., 2021).

Raw data coming from the payment system will include features such as time, amount, and place of the transaction (Correa Bahnsen et al., 2016). These features occur in a single transaction and, therefore, fail to represent customers' behavior. Fraud detection requires features that characterize users based on their behaviors (Kou et al., 2004).

APATE (Van Vlasselaer et al., 2015) proposes the creation of features based on the consumer spending history, taking inspiration from the Recency-Frequency-Monetary value (RFM) model and combines it with time-dependent suspiciousness scores obtained from the payment network objects (merchants and credit card holders).

Another approach uses Hidden Markov Model to create sequences of transactions modeled to represent behavior aspects (such as the spent amount and the time elapsed since the previous transaction). The likelihood of each of these sequences is calculated and employed as a synthetic feature for fraud detection (Lucas et al., 2019).

Autoencoders are also proposed (Fanai & Abbasimehr, 2023) as a feature engineering technique due to their capacity to find and represent patterns in the data.

2.5 Models used for Fraud Detection

2.5.1 Supervised models

The description of the models is presented in the Theoretical Background chapter.

Supervised learning is used when there is a dataset for which a target, or label, is known. It assumes that patterns will arise from the analysis of historical data and that using those patterns, the model employed will be able to predict the label for new, unseen data.

Fraud detection is usually considered a supervised learning problem (Leite, Miguel Lobo Pinto, 2020), in which the purpose is to determine whether or not a transaction that is being processed is fraudulent (positive label) or legitimate (negative label).

Many distinct supervised models are studied and employed in fraud detection (Tiwari et al., 2021), including Logistic Regression, Decision Trees, Random Forest (Mirhashemi et al., 2023), Hidden Markov Model (Hegde et al., 2022), Support Vector Machines (Murugan et al., 2023), Artificial Neural Networks (Rb & Kr, 2021), Deep Learning (San Miguel Carrasco & Sicilia-Urbán, 2020), Bagging (Zareapoor & Shamsolmoali, 2015) and several model ensembles (Khalid et al., 2024).

Logistic Regression creates a linear model, to which a function such as the sigmoid is applied to force the values to be between 0 and 1, with the label (True/False) being determined by comparing this result with a threshold, often 0.5, but that can be any other valid number.

Decision Trees are an easy model to implement, flexible in the treatment of data, and provide results that are relatively easy to explain. All of these factors make them quite popular. They do tend to overfit (Omair & Alturki, 2020), meaning that they perform poorly on unseen data. This can be partly compensated, for instance with the adequate pruning of some nodes. Some tree-based models address the weaknesses of using a single decision tree, while retaining some of its best characteristics and performance (Grinsztajn et al., 2022), as is the case of Random Forest, LightGBM and XGBoost.

Hidden Markov Model (HMM) can learn to identify legitimate transactions done by the cardholder, and encode that information in customer profiles (Lucas et al., 2019). Besides being used for feature engineering, HMM can be used for predictions. The performance of the model is very dependent on the existence of profile information, and when the difference between front and non-fraud is slight, the performance is degraded by a decrease in True Positives or an increase in False Positives (Tiwari et al., 2021).

Support Vector Machines try to find the hyperplane in an n-dimensional space that provides the best separation (meaning, creates the largest possible margin) between two classes. To further facilitate this separation, different kernel functions are tested during the train phase (the search for the hyperplane that provides the best separation), and the best kernel function is then used (Murugan et al., 2023).

Bayesian Belief Networks are based on Bayes' conditional probability theorem; conditional probabilities are calculated for each class, and each element is considered to be part of the class with the highest conditional probability (Tiwari et al., 2021).

K-Nearest Neighbors predicts the class (fraud/not fraud) for each data point based on the class for the k data points that are nearest to the point to predict. Besides determining the number of neighbors to consider, with lower numbers generating more overfitting, it is mentioned that because this is a lazy algorithm, there is no need to train the model, but the time to make a prediction is higher, as there is the need to find the closest neighbors (Sultana et al., 2023). Even though this model is reported as having good performance under some conditions, it is not practical for scenarios where a real-time prediction is expected.

Artificial Neural Networks (ANN) are inspired by the human brain, replicating the neurons and the synapses that interconnect them (Vanneschi & Silva, 2023, p.161). In a simple ANN each neuron receives input signals from one or more neurons in the previous layer, transforms these inputs based on its weight parameters and bias, applies a function and outputs the result to one or more neurons at the next level. By learning from train data (data whose label – fraud/not fraud – is defined), the network neurons adjust their weights/bias, so that when given new data, the model will be able to predict the label.

Deep Learning, including Convolutional Neural Networks and Recurring Neural Networks, further develops basic ANN's by adding more layers or allowing for the transfer of information

between non-sequential layers. It is also used in fraud detection (Shenvi et al., 2019; Hassan et al., 2020).

Base models such as the ones mentioned can also be combined; these combinations are referred to as model ensembles. A model ensemble tries to make better predictions by combining several models, benefiting from this diversity and from specific advantages of some base models. Ensembles have also been implemented for FDS (Khalid et al., 2024).

Bagging, an abbreviation of "Bootstrapping and Aggregating" is an ensemble that combines the predictive power of several models in order to obtain a prediction that is better and more consistent than each individual model's predictions. Bootstrapping is the process of random choice with repetition of observations to use as the train dataset. This ensures a reduction in overfitting. Each base model is trained and makes its predictions independently, making it possible for this processing to be parallelized. Aggregating is the combination of these predictions, using some criteria. For binary classification problems, this is usually obtained by majority voting (final prediction is the class predicted by mode models), weighted voting (the weight of each model's prediction is based on that model's performance), or probability-based voting (instead of presenting a class label, each model will inform the probability assigned to each class; the final vote is the class with a highest average probability).

Boosting uses several models, but each model provides to the next model information on what data points were incorrectly predicted. In boosting each model will assign more weight to the predictions that were missed by the previous model. This in fact increases the cost for failing in those same observations, and so favors models for which these labels are being correctly predicted. This architecture prevents this ensemble from being parallelized, as each model is dependent on the results from the previous model.

Stacking is another ensemble architecture. While bagging and boosting are typically implemented with weak base models and obtain their predictive capacity from the use of a large number of these models, on the contrary stacking uses a small number of expert models to make independent predictions. Then these predictions are fed as input into a separate model (the "meta model") that will make a final prediction.

Tree-based ensembles, such as the ones presented below, are very common in Fraud Detection Systems, as they provide very good results using tabular data and offer better explainability than most alternatives.

Random Forests consist of groups of decision trees. While maintaining the Decision Tree's ease of implementation and flexibility in handling data, by growing several independent trees with bootstrapped data and with a randomized subset of features, Random Forests reduce overfitting. Each decision tree generates an individual prediction, and these are then combined to create a final prediction. In many circumstances, Random Forests emerge among the best-performing models. These advantages make them useful, and used, in FDS contexts (Santos, 2020).

Gradient Boosting Decision Trees (GBDT) is a combination of Boosting with Decision Trees. By training each tree on the same data but assigning a higher focus to the samples that were not correctly classified, this approach is capable of outperforming other models and ensembles. One of the implementations is Extreme Gradient Boosting (XGBoost). It provides competitive results in some configurations, but struggles in terms of scalability and efficiency when dealing with datasets that have a large number of features or observations, due to the complexity of calculating the information gain at each possible split.

LightGBM (Ke et al., 2017) is a popular implementation of GBDT that, by making reasonable assumptions on two dimensions (Gradient-based One-Sided Sampling or GOSS and Exclusive Feature Bundling or EFB), significantly reduces the computational cost, yet with a very good approximation to the ideal splitting point. LightGBM is a model frequently used for fraud detection (Santos, 2020).

2.5.2 Unsupervised models

Unsupervised learning uses unlabeled datasets, meaning that the dataset has no target variable. The entries in the dataset are processed with a certain objective, which often is the organization in clusters.

A cluster is a group of entries that are similar to each other, but different from entries in other groups. Clustering aims to find the groups that maximize the similarity within the group while maximizing the dissimilarity between elements in different groups. Clustering is well-known in business areas such as marketing, as it allows for the creation of segments of customers, enabling campaigns that are targeted at clients that are relatively similar to one another.

Behavioral outlier detection is another unsupervised fraud detection technique (Bolton & Hand, 2001), and it can identify anomalies in a transaction. These anomalies are relevant for fraud detection (Carcillo et al., 2021).

Other unsupervised techniques include using graphs to determine nodes that are close to each other or that are distant but similar in structure (Li et al., 2023).

2.5.3 Semi-supervised models

Supervised learning requires labeled data, which may be scarce, expensive to acquire, and is typically obtained *a posteriori*. So, it is basically learning from the past. Unsupervised learning can benefit from an abundance of unlabeled data, and can find both clusters and outliers, which is useful information to determine fraud. By finding outliers, it is hopefully possible to capture a new type of fraud, something that, for supervised learning, would be hard to do, as it relies on past data.

Semi-supervised learning aims to use the two complementary approaches and is commonly used when there are many unlabeled and few labeled data points (Carcillo et al., 2021).

Active Learning (AL) addresses the issue of having abundant data but label scarcity, by selecting for manual labeling the instances that are considered to be most informative. These will be the minimum number of entries necessary to train the model that will be employed (Krishnakumar, 2007).

These requirements closely match the reality of fraud detection, and in fact, AL has been used for this purpose (Leite, Miguel Lobo Pinto, 2020; Carcillo et al., 2018).

2.5.4 Nature-inspired models

> *"Artificial immune systems (AIS) can be defined as computational systems inspired by theoretical immunology, observed immune functions, principles and mechanisms in order to solve problems" (de Castro & Timmis, 2003).*

A vertebrate's immune system faces challenges that can be mapped to an AIS used for fraud detection (Soltani Halvaiee & Akbari, 2014): there is a need to separate normal from unauthorized occurrences; the number of normal occurrences is much higher; normal and unauthorized elements are quite similar, as fraudsters/viruses try to emulate body cells/legitimate transactions; frauds that are previously unseen need to be detected, and should be memorized for future recognition.

For fraud detection, other techniques employed include optimization inspired by animal behavior, including variations on Particle Swarm Optimization, such as Group Search Firefly Algorithm (Jovanovic et al., 2022).

2.6 HYPERPARAMETER TUNING

Part of any machine-learning project is the hyperparameter tuning, which consists in finding the best configuration for the model, given the specific problem that is being addressed. For a Random Forest it will include configurations such as the number of instances, the minimum number of data points per node or the maximum depth; for a neural network it may include the number of neurons per layer and the learning rate.

The hyperparameters available for each model, and the corresponding domain of values, constitute a search space. Hyperparameter tuning tries to find the global optimum in that search space.

Manual search might be considered the most elementary option for hyperparameter tuning and involves trying several parameter value combinations and recording the one that yields the best results.

Automated approaches include basic techniques such as grid search and randomized search (Bergstra & Bengio, 2012). Grid search, given a hyperparameter search space, will try all possible combinations of values for all hyperparameters in the search space. This is a brute force approach that, depending on the number of parameters and the domain of values

defined for each parameter, might create a gigantic search space, resulting in an optimization process that theoretically ensures the best result possible, but in practice, will take longer than the acceptable amount of time. An alternative is randomized search, which limits the trials to a predetermined number and, for each attempt, will consider randomly chosen values for each parameter, as defined in the corresponding search space. Obtaining a good result in an attempt provides no information for the next attempt, which will be totally random. It is possible to obtain results quicker than with Grid Search, but there is no longer the guarantee that we will obtain the best possible hyperparameter values. In fact, this will dependent on luck in the randomized parameter value selection (Anggoro & Mukti, 2021).

Due to the inefficiency of these approaches, other methods for accelerating hyperparameter tuning have been proposed, such as successive halving algorithms (Soper, 2023). This approach optimizes parameters in several successive iterations, and at each iteration, the number of models reduces exponentially while the number of data points to train increases exponentially. Successive halving starts with many models and a small dataset and moves on by significantly decreasing the number of models (by keeping the best-performing at the previous stage) and increasing the size of the dataset until the best configuration is found. This does not ensure the global optimum when using the whole dataset, as conceivably, that global optimum might have been rejected when it was evaluated with a smaller subset of the data.

Bayesian optimization abstracts hyperparameter as an optimization problem that can be represented and solved as a Gaussian process (Snoek et al., 2012), attaining results that compare favorably with human expert-level optimization.

2.7 METRICS

Given that fraud detection is, in nature, a supervised learning problem, specifically a binary classification problem, we can, in principle, analyze FDS performance using standard measures such as accuracy, sensitivity, precision, and specificity (Adepoju et al., 2019).

Fraud detection systems that work well in a controlled laboratory setting may be unsatisfactory in terms of business (Provost & Fawcett, 2013, p. 31) because even if the model proves to have very high accuracy, it might still produce a large number of false positives, making it economically inviable due to the resulting costs of dealing with those false alarms and the significant cost of customer dissatisfaction.

However, we should consider that fraud detection is a cost-sensitive problem (Correa Bahnsen et al., 2016). The cost for false negatives is quite direct - the amount of that transaction is lost (Hand et al., 2008). Depending on the service contracted, the entity that will ultimately incur this cost might be the end-user, the merchant, the acquirer (bank or other), or an insurance company. We will consider that there is a cost, regardless of who gets penalized.

For false positives, the payment network faces the tangible cost of lost revenue in transaction fees, and the merchant faces the cost of the actual transaction that was not processed.

However, there is also a less tangible cost (administrative costs, reputation costs, and potential dissatisfaction from the legitimate customer).

Furthermore, we cannot assume a constant cost for each type of false prediction because both genuine and fraudulent transactions vary significantly in terms of the amount. To address this, cost-based measures can be employed to approximate the actual costs for each case and for each false prediction (Correa Bahnsen et al., 2016).

3 THEORETICAL BACKGROUND

This chapter introduces concepts, techniques, models, and metrics relevant to the analysis conducted in this thesis. The content explained in this chapter is also used in the methodology and experimental study chapters. These are the tools that will be used to build the proposed methodology for evaluating the performance of predictive models for fraud detection.

3.1 MACHINE LEARNING PIPELINE

3.1.1 Motivation for using an ML pipeline

The typical supervised learning project involves several steps that are executed sequentially. Data must be collected, cataloged, combined, cleaned, pre-processed, filtered, and augmented. Only then can it be presented to a supervised learning model that will assess the data and, by discovering patterns and representing the reality of that data, learn how to predict the class to which new observations belong.

The system's predictive capability depends on the operations performed on the data and on their sequence. This combination is referred to as a Machine Learning pipeline. By changing the steps or their sequence, a new and unique pipeline is constructed.

The evaluation of two distinct pipelines will tell us which one performed better but will not discriminate which steps increased the predictive capability and which ones didn’t.

3.1.2 Components of an ML pipeline

The exact composition of a Machine Learning pipeline will depend on the Data Scientists studying the problem. A pipeline is usually composed of a sequence of preprocessing steps followed by a supervised model (Müller & Guido, 2016, p. 310). The first steps will import and transform the data, the last step (the model) will be responsible for making predictions. For binary classification problems such as fraud, the prediction will be “positive” (fraud) or “negative” (legitimate).

Even though the first step will always be data ingestion, different data sources may be used, and therefore the output of this first step will vary. And even though the last step is the supervised learning model making predictions, different models may be employed, meaning the results will differ.

Between these two extremes, not only may each step have more than one option, but steps can also be added, removed, or reordered differently. All these possibilities may convey the idea that comparisons will be hard. It should nevertheless be clear already that the pipeline can be seen as a system that can learn from train data and make predictions.

The rest of the chapter addresses some of the key options available for different pipeline steps and for measuring results.

3.2 OUTLIER HANDLING

3.2.1 Motivation for handling outliers

Outlier handling is part of a typical Machine Learning pipeline.

An outlier is a data point that appears to be inconsistent with the remainder of the set of data in which it is observed (Barnett & Lewis, 1998). An outlier has implicitly associated the idea of being distant from other observations.

> *"Outlier detection has been used for centuries (…). Outliers arise due to mechanical faults, changes in system behavior, fraudulent behavior, human error, instrument error or simply through natural deviations in populations. Their detection can identify system faults and fraud before they escalate with potentially catastrophic consequences. It can identify errors and remove their contaminating effect on the data set and as such to purify the data for processing." (Hodge & Austin, 2013).*

These extreme observations may negatively impact the model's performance, either by impacting other preprocessing tasks or by directly impacting the model.

First of all, it is necessary to identify outlier observations. Then a decision has to be made on the best course of action to minimize their negative impact.

3.2.2 Univariate outliers

A univariate outlier is a data point for which one feature registers a value that significantly deviates from the distribution of values for that feature. This may be a feature that follows a heavy-tailed distribution or a feature generated from a different, "contaminating distribution" *(Hawkins, 1980)*. An example would be a payment of an extremely high or low amount.

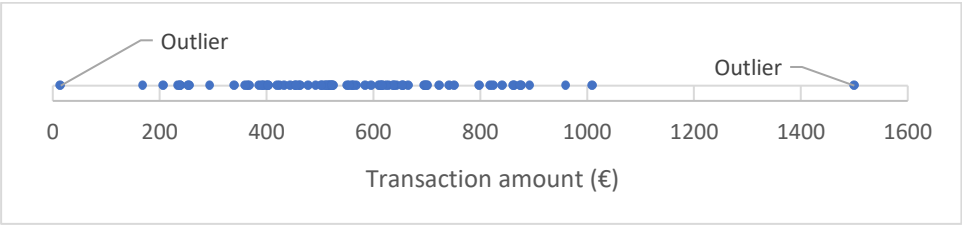

Figure 3.1 - Univariate outliers for feature "transaction amount"

This type of outlier can easily be detected using statistical distributions for each feature. It is common to consider as outliers values below 1st Quartile - 1.5 x IQR or above 3rd Quartile + 1.5 x IQR, with IQR being the interquartile range, which corresponds to the difference from the 1st to the 3d quartile *(Han et al., 2012, p. 50)*.

3.2.3 Multivariate outliers

A multivariate outlier may have values that are common for all of its features, and yet present an uncommon combination of values for some of those features.

Considering a dataset that has transactions such as money transfers, credit card payments, and mobile payments, a transaction of 600 euros might be common for money transfers and credit card payments but uncommon for a mobile payment. In this case, what makes the observation an outlier is not the value for either feature (mobile payments are common, payments of 600 euros are also common); it's the combination of both that is uncommon.

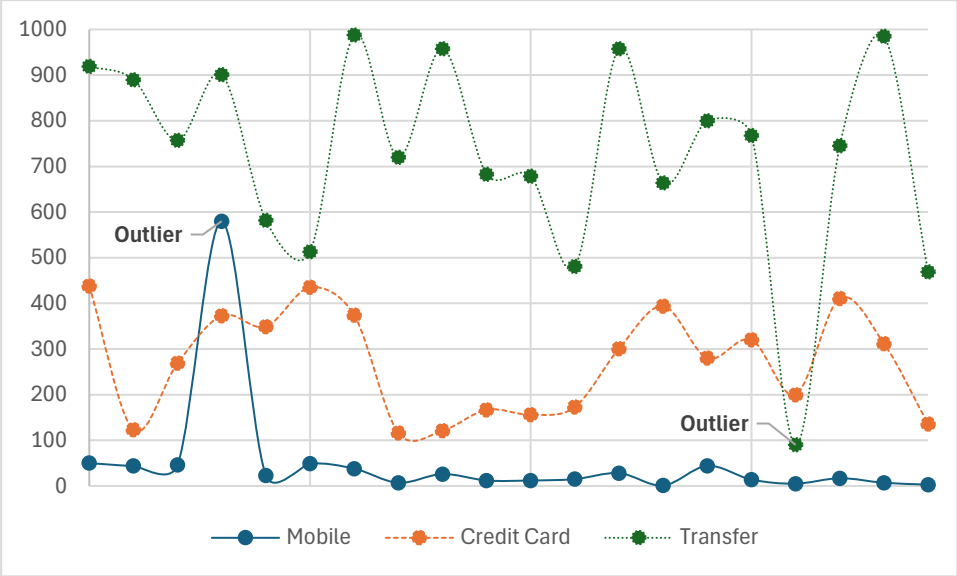

Figure 3.2 - Multivariate outliers, features payment amount and type

This type of outlier cannot be found by looking at just one feature; there is the need to use some technique that measures the distance between observations, either directly or indirectly through clustering or density-based techniques *(Acuna & Rodriguez, 2013)*. Data Mining techniques such as clustering can be used to find these outliers, and in fact, some techniques (e.g., DBScan) directly determine and inform which data points are outliers.

3.2.4 Impact of Outliers on Models

The impact of outliers on a model depends on the number of outliers (the more outliers, the larger the impact), the size of the dataset (the larger the dataset, the smaller the impact), and how distant from the other points the outlier is (an outlier that is orders of magnitude larger than the remaining observations will have more impact than one that is larger by a smaller margin).

Different models suffer different impacts from the presence of outliers.

Decision Trees are quite robust to outliers, meaning that inserting or removing outliers has a neglectable impact on the model. This results from the fact that decision trees do not consider

distances between points but rather compute the “information gain” (Bramer, 2016, p. 54) from each split. What is relevant is which data points are placed on each side of the split. The same is valid for tree-based models (models that combine Decision Trees in some way, such as Random Forests and XGBoost).

All models that compute or measure distances between data points, such as linear and logistic regression, artificial neural networks, and support vector machines, are significantly affected by outliers.

To demonstrate this effect, below we see the data that was represented as “Mobile” in the previous figure. A Linear regression that fits the data is drawn, and then a new linear regression is drawn after removing the outlier. It is visible in the next figure that both the intercept and the slope of the linear regression change significantly depending on whether the outlier is kept in the data set.

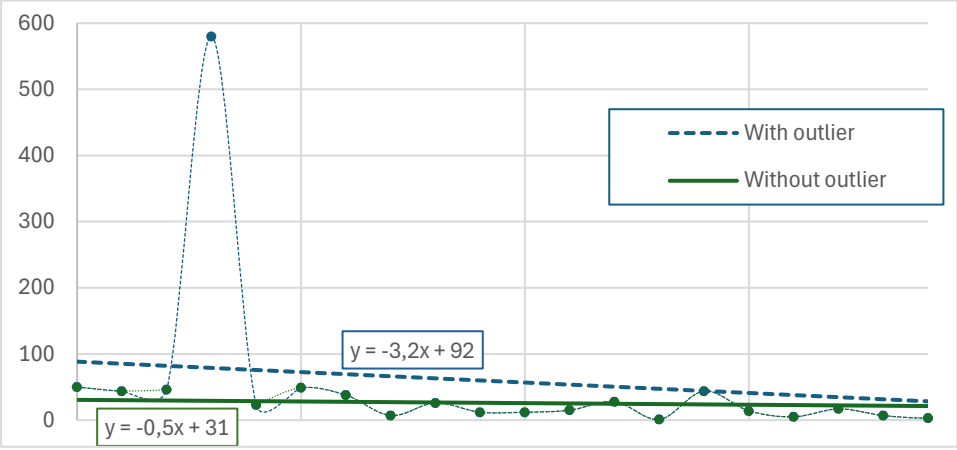

Figure 3.3 - Impact of keeping/removing an outlier

3.2.5 Handling outliers

Outliers can be handled in several ways. In the train dataset, they can be ignored, updated to values closer to the feature's distribution, updated to Null, or even removed. In validation and test datasets, removing data is not a valid option, and if updating, it must be assured that no data leakage occurs.

Besides its value as observations, outliers can be used for other purposes such as feature engineering, by marking them as anomalies. Anomaly detection is very useful for fraud detection and for intrusion detection, which makes it very well suited for scenarios of extremely imbalanced datasets (Thimonier et al., 2023).

3.3 DATA SCALING

3.3.1 Motivation for scaling data

Numerical features in a dataset typically vary in amplitude. For example, considering a dataset with information on Bank clients, the feature Debt-to-income (DTI) ratio may usually vary between 0 and 0.5, while the average transaction amount could vary between 50 and 1000.

Let us consider three customers with a specific set of values in each feature and measure the distance from each customer to customer #1. Customer #2 is significantly farther away from customer #1 than customer #3. In fact, we can see intuitively that customers #1 and #2 are very similar. However, they are different in a feature that has a very large amplitude (0 to 1.000). Contrary to that, customer #3 seems to be extremely similar to customer #1 when, in fact, they are different by one order of magnitude in one feature; it just happens to be a feature whose values are in a very small interval (0 to 1).

Table 3.1 - Information on three customers, unscaled

Customer	Average transaction	DTI ratio	Distance to #1
#1	1.000	0.1	0
#2	950	0.1	50
#3	1.000	1.0	0.9

Features are scaled so that, by assigning a similar range to different features, the measurement of distances will, in fact, be a valid indication of the similarity of the observations whose distance is being measured.

If we divide each feature by the maximum amount (note that this is not an actual technique and is used just to demonstrate the concept), the distances measured will now adequately express the fact that customers #1 and #2 are very close to each other and very distant from customer #3.

Table 3.2 - Information on three customers, scaled

Customer	Average transaction (scaled)	DTI ratio	Distance to #1 (scaled)
#1	1	0.1	0
#2	0.95	0.1	0.05
#3	1	1.0	0.9

As with outliers, the impact of scaling depends on the type of model. Tree-based models are expected to be less sensitive to scaling than models that heavily rely on measuring distances or gradient descent, such as Logistic regressions, artificial neural networks, and Support Vector Machines.

3.3.2 Scaling data without leaking information

Scaling data in a feature involves applying a formula to each occurrence of that feature. The formula must be established using solely data in the train dataset, so if the formula involves determining statistics such as the minimum, the mean, or a percentile, those will be calculated using exclusively train data. This analysis is called fitting the data and, as mentioned, is only performed in the train dataset.

After the formula is established for a certain feature, it will be applied to the train, validation, and test datasets. This operation is the actual transformation of data, and it is done with the exact same formula that was established on the train data.

As an example, consider we're scaling a feature using a formula that uses the minimum, which is found to be 2. This minimum must be obtained using exclusively the train dataset, regardless of the fact that in validation and test, there may be smaller values. Moreover, the value obtained from the train dataset will be employed for scaling validation and test data.

A consequence of this is that when a certain transformation places all values of a feature between 0 and 1, this is only guaranteed for the train data. The resulting formula may cause observations in the validation and test datasets to be lower than 0 or higher than 1. This is so by design; the formula must not be "re-calibrated" for the data in the validation and test dataset. If there is the need to keep the scaled value inside that interval, even for validation and test datasets, it is possible to set values above the upper limit or below the lower limit to the corresponding boundaries (Bramer, 2016, p. 35).

3.3.3 Min-Max scaler

Min-Max scaling consists in applying a linear conversion to all values in a feature so that the scaled results will be within a certain range, which typically is [0,1].

If X is the value to be converted, X_{minTr} and X_{maxTr} are the minimum and maximum for that feature in the train dataset, the formula to scale X_i to the range [0,1] is:

$$X_{scaled} = \frac{X_i - X_{minTr}}{X_{maxTr} - X_{minTr}}$$

Other ranges can be used, such as [-1,1]. The formula above is a particular case for the interval [0,1]; for a generic range [low, high], the formula becomes (Han et al., 2012, p. 114):

$$X_{scaled} = \frac{X_i - X_{minTr}}{X_{maxTr} - X_{minTr}}(high - low) + \text{low}$$

As mentioned, the interval is defined for data in train; it is possible that the scaled values for validation and test data will be outside this interval.

This scaler is sensitive to outliers.

3.3.4 Standard scaler

Standard scaler is also known as z-score scaler, or zero-mean scaler. It should preferably be used for data that follows a Normal distribution (Izonin et al., 2022).

Features are standardized by subtracting the mean and dividing by the standard deviation (Qi et al., 2023). Representing the mean of the feature in the train data with μ, and the standard deviation with σ, the scaling for observation X_i is done using the formula:

$$X_{scaled} = \frac{X_i - \mu}{\sigma}$$

As before, the formula is fixed on the train dataset and applied to the train, validation, and test datasets.

This scaler uses the mean and the standard deviation in calculations, making it also sensitive to outliers.

3.3.5 Robust scaler

Robust scaler addresses the need for a scaler that is more robust to outliers (Izonin et al., 2022).

The statistics used for centering the data and for scaling are based on percentiles and are, therefore, less influenced by a limited number of outliers. This scaler assumes that the data being transformed follows a normal distribution.

Representing the interquartile range as IQR and median(X) as the median for feature X, the scaling for observation X_i uses the formula:

$$X_{scaled} = \frac{X_i - median(X)}{IQR}$$

3.4 NULL VALUE HANDLING

3.4.1 Motivation for null-value handling

Null represents the absence of value. This can result from a condition in which the value does not apply (e.g., feature “authorization_elapsed_time” is Null because the transaction was not authorized) or that was not recorded due to some human/machine failure (e.g., a transaction was authorized, but the system failed to record the time it took to obtain the authorization).

For our purposes, it is highly undesirable to have null values, as they will affect calculations and even render some techniques and algorithms uncapable of operating.

3.4.2 Drop null values

An option that may be considered is to drop null values. This can be done by removing the observation (row) or by removing the feature (column). The option to remove the observation is only possible in the train dataset; the strategy for validation and test dataset cannot involve removing observations, as we need to make a prediction for all observations.

3.4.3 Use a statistic or predefined value

For numerical and metric features, common techniques include replacing the null value with a predetermined value, which might be a previously-set constant or a statistic (such as the mean or the median) that is calculated on the train dataset and then is used for train, validation, and test datasets.

For features that are numerical but not metric, statistics such as mean and median do not have an intuitive interpretation. In these cases, null values may be replaced with the mode. This is also a common approach for categorical or binary features.

3.4.4 Use nearest neighbors to assign a value

The approaches described so far can somehow be compared to the univariate outlier detection in the sense that they rely on one single feature.

K-nearest neighbors is a more sophisticated approach. It may be compared to multivariate outlier detection, in the sense that it uses several features. This procedure involves determining the group of K data points that are closer, based on all the remaining features, and then using that neighborhood to determine the value to be assigned.

The rationale is that an observation that is very similar to several others in many characteristics is also likely to be similar in the unknown characteristic (null). To provide some added robustness and improve generalization, several neighbors are considered, not just one.

To determine the value to assign to one feature for one single data point, it is necessary to calculate its distance to all the other data points in the dataset. For large datasets and datasets with many null values, this becomes unfeasible.

3.4.5 Presence of Null as a new feature

In a feature with a significant percentage of Null values, it is intuitive that the feature, not the observation, should be discarded.

Alternatively, a new binary feature might be created just to record whether or not the value was Null. If the Null value in the “authorization time” results from some malfunction on a part of the system, and this malfunction is being exploited for fraud, then having a feature merely

stating whether or not “authorization time” was measured might in fact convey information that can be used by the model.

3.5 FEATURE ENGINEERING

3.5.1 Motivation for feature engineering

Feature engineering consists in creating new features through the manipulation of one or more existing features. E.g., from the amount of a transaction we can create a feature to compute the logarithm of that amount; another new feature can be obtained by dividing the amount by the account balance. Features can also be obtained by looking at other observations, e.g., create a feature to represent the number of seconds since the previous transaction.

According to Abedin et al. (2023):

> *One of the most important elements of bank customer classification models is feature engineering, which is the process of transforming raw data into new features and selecting the best features for improving the performance of classification methods. This often overlooked aspect is becoming increasingly important, especially in the contemporary world of financial uncertainty caused by the pandemic. In fact, previous classification models (F. Aslam et al., 2022; Y. Liu et al., 2022; Yuan et al., 2022) tended to neglect the importance of feature engineering*

It is known that Machine Learning uses mostly secondary data, meaning that it was collected for some operational purpose and then also made available for analysis. Attributes on the raw data set may not be relevant as they were initially created but may become interesting after some transformation. E.g., the attribute “date of birth” might not be a good attribute for a Machine Learning algorithm, but the attribute “current age” calculated from the date of birth may be extremely relevant.

3.5.2 Principal Component Analysis (PCA)

Data points in a dataset with K features can be represented in space of K dimensions, with each dimension corresponding to a feature and the value for that feature corresponding to a coordinate along that axis.

Principal Component Analysis (PCA) is a mathematical method that performs an orthogonal transformation in this space. Principal component 1 (PC1) will be an axis defined so that it captures the largest possible variance in the dataset. Principal component 2 (PC2) will be orthogonal to PC1 (Destefanis et al., 2000) and will capture the largest amount possible of the remaining variance. In the end, the number of principal components will be equal to the number of initial features, with the difference that the former will be ordered from the most significant to the least significant, in terms of variance of the dataset.

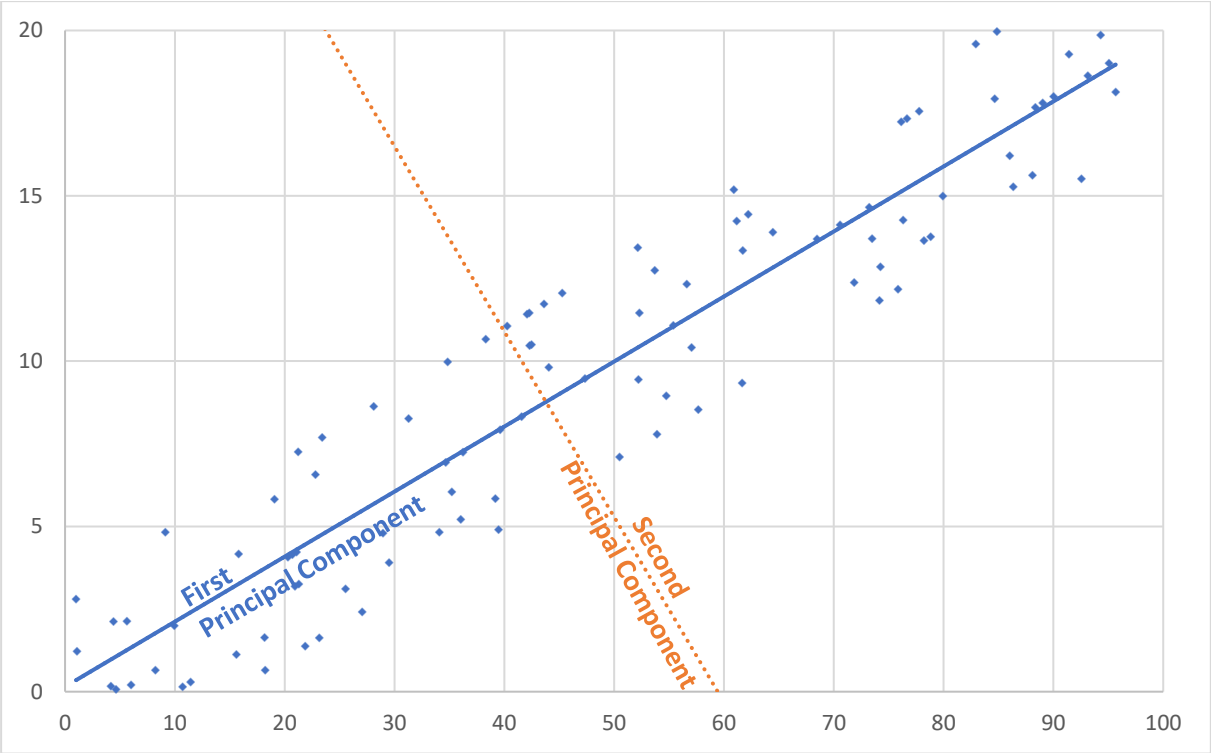

Figure 3.4 - Principal Component Analysis (PCA)

This general technique can be used for several purposes, and is often mentioned for feature reduction; new features are created, only the most significant are kept, and all original features are discarded.

For the purpose of this thesis, it is more interesting to include features generated through PCA in the original dataset, and allow for the models to take advantage of the combination that is present in each of these components.

Each component created can be expressed as a linear combination of all the original features in the dataset, in which the coefficients are independent. This means that decisions made over a PCA feature will be extremely hard to explain or visualize in terms of the original features, because in fact all of them are involved, to a certain degree.

3.5.3 Linear Discriminant Analysis (LDA)

For classification problems, Linear Discriminant Analysis (LDA) identifies the line that maximizes the separation between the classes. Data points are then projected on a line that is perpendicular, therefore maximizing the separation of the classes. LDA attempts to perform a transformation that maintains the class structure that is present in the original higher-dimensionality space (Ye et al., 2004), but on a lower-dimensionality space.

As was the case for PCA, features created using LDA are also a linear combination of the features in the original dataset.

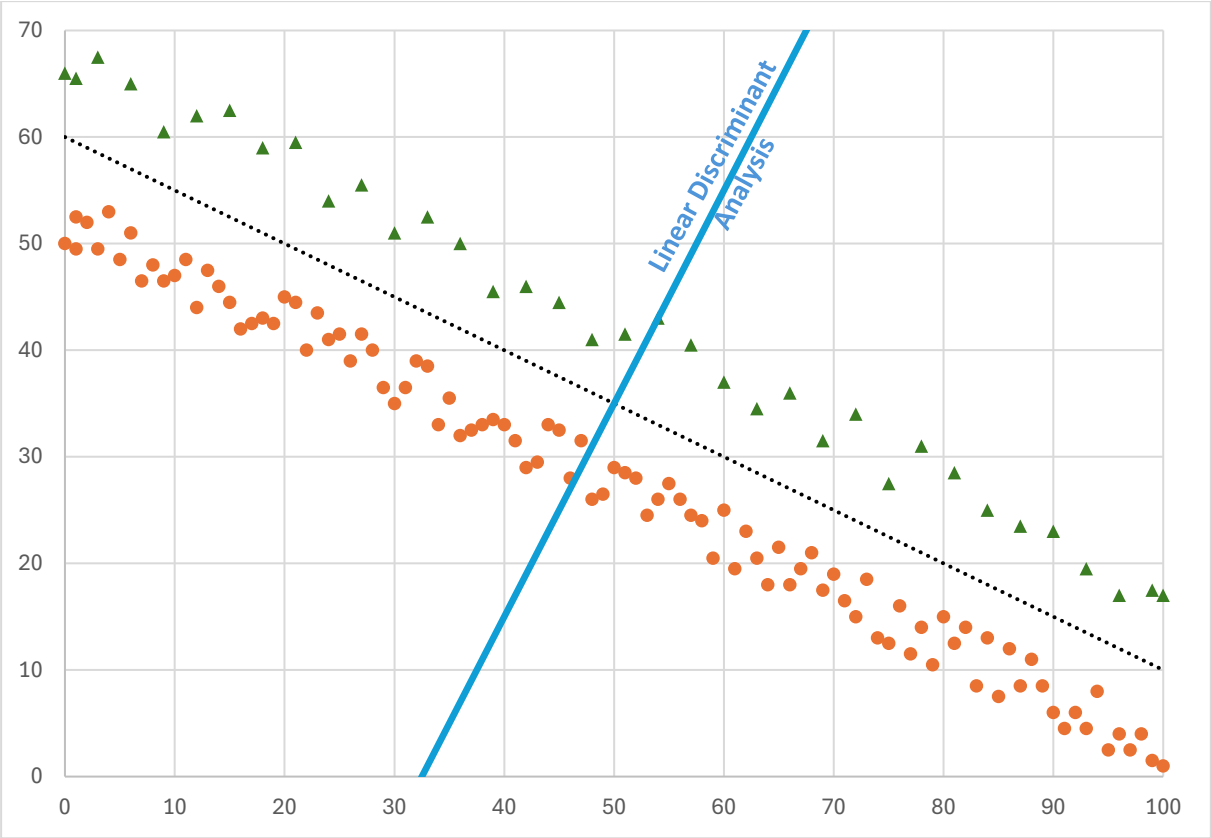

Figure 3.5 - Linear Discriminant Analysis (LDA)

For a multiclass problem with n classes, this approach can produce a maximum of n-1 new features. In particular, for binary classification problems such as fraud detection, it is only possible to produce one of such lines.

3.5.4 Expert analysis and AutoML

A data scientist with extensive field knowledge may identify features that can be engineered from the existing dataset, and which may contribute significantly to the predictive capabilities of the model. This group of new attributes created from expert opinion could be implemented as an additional option in the DOF that corresponds to feature engineering. These new attributes would be included if and only if that option was the randomly selected one for that DOF, being added or not, depending on the randomly selected choice.

In the end, this option was not made available for the feature engineering DOF, as it was perceived that the results could be biased due to different levels of knowledge in different datasets.

Strategies involving brute force experimentation in feature engineering were also discarded, as it was anticipated that this approach would not yield actual insights.

The area of AutoML, by trying to create complete, working pipelines with minimal expert involvement, can contribute to the process of feature engineering. While this was also not implemented, it may merit further investigation.

3.6 FEATURE SELECTION

3.6.1 Motivation for Feature Selection

Feature selection consists in discarding attributes in the dataset that appear to be irrelevant (Russell et al., 2022, p. 689) or that have a negative impact on models, as is the case of overfitting. On large datasets with high dimensionality, feature selection is used to improve the performance of the estimators, to make models run faster and in a more cost-effective manner, and to allow for a better understanding of the process that is generating the data (Guyon & Elisseeff, 2003).

There are several techniques and statistical tests for determining which features to remove and which features to keep. Some tests are performed on a single feature, others on a group of features. These tests are used to infer the relevance of the feature for the model.

Feature selection methods can be organized into three groups. Filter methods require no model involvement; features are assessed using statistical techniques that may be univariate or multivariate and considering the relation of the feature with the target variable. Wrapper methods address the feature selection process as a search problem, then evaluate and compare combinations of different sets of features with the assistance of a predictive model. Finally, embedded methods determine the best features while the model is being trained, typically by assigning a penalization for more complex models, causing extra features to only be included if their benefit outweighs the penalization incurred.

For increased robustness, several tests can be conducted, with the decision of keeping or removing a feature coming from a majority vote or some other form of voting.

3.6.2 Null percentage

This is a univariate filter method; it calculates the percentage of Nulls in each column and compares it against a defined threshold; columns above that threshold are candidates to be removed.

3.6.3 SelectKBest

SelectKbest is a filter method that selects the best features in a dataset, based on their performance. This performance is assessed using a score function, such as an f-test. The f-test is a statistical test whose null Hypothesis is that a group of explanatory variables has no effect on the dependent variable (Wooldridge, 2019, p. 139). If we fail to reject this hypothesis, then those explanatory variables (features) are candidates to be removed from the dataset.

This method has proven capable of outperforming other feature selection techniques (Saeed & Hama, 2023).

3.6.4 Chi-squared

This method calculates, for each feature, the chi-squared statistic with respect to the classes of the target variable (H. Liu et al., 2002). The features are then sorted in descending order of the chi-squared statistic, resulting in the features being now ordered by importance, from the highest to the lowest.

3.6.5 Pearson correlation

For each pair of columns, compute the correlation using the Pearson correlation coefficient. This coefficient is obtained by dividing the covariance of the two variables by the product of their standard deviation, as shown in the formula:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)}{\sigma_X \sigma_Y}$$

This value presents a normalized covariance, varying in the interval [-1,1].

3.6.6 Spearman correlation

Spearman's rank correlation measures the statistical dependence of the ranks of two variables. It corresponds to the Pearson correlation not of the variables but of their ranking. Spearman rank correlation assesses monotonic relationships but does not require linear relationships, as does Pearson correlation.

Spearman correlation also varies in the interval [-1,1].

3.6.7 Kendall's Tau coefficient

Kendall's Tau coefficient presents the ordinal association between two quantities, by measuring the similarity of the data when ranked by each of the measures. Two occurrences (x_1,y_1) and (x_2, y_2) are concordant if x_1 and y_1 are either both greater than or both smaller than x_2 and y_2. Otherwise, they are discordant.

$$\mathrm{T} = \frac{(numberofconcordantpairs) - (numberofdiscordantpairs)}{(numberofpairs)}$$
$$= 1 - \frac{2.(\mathrm{numberofdiscordantpairs})}{\binom{n}{2}}$$

Kendall's Tau coefficient is close to 1 when a large number of pairs is concordant, and close to -1 when a large number of pairs is discordant.

3.6.8 Mutual information

Mutual information, or information gain, is a positive number that measures the amount of information that can be obtained about the target variable by observing the independent variable.

If the value of mutual information between an independent feature and the target variable is small, this indicates that this variable contains no or very little information about the target variable, and therefore it is a candidate to be dropped from the dataset.

3.6.9 Recursive Feature Elimination

Recursive feature elimination is a wrapper method. It employs an estimator model that is trained on all features, and then determines which of these is the least significant. This feature is then removed, and the process is repeated, until a certain pre-determined number of features is obtained.

By recursively eliminating the features that are least significant, we identify the reduced set of features that contribute the most to the predictive capabilities of our model.

3.6.10 Least Absolute Shrinkage and Selection Operator (LASSO)

Lasso is an embedded method. This means that it learns which features contribute to the accuracy of the model while it is being created. It introduces a penalization on more complex models, so that the results are biased towards less complex models (those that have less features and, therefore, less coefficients).

3.7 DATA IMBALANCE

3.7.1 Motivation for Data Imbalance handling

A dataset is imbalanced when the number of observations belonging to each class of the target variable differs significantly. The focus will be on binary classification problems, but the approaches presented remain valid for target variables with more than two classes.

A model trained on an imbalanced dataset may become biased towards the majority class and fail to capture the structure of the observations in the minority class. Class imbalance is a very common real-world situation for most types of problems, and it is noteworthy that usually, we are more interested in the prediction of the minority class (be it a fraud, a rare disease, or an oil spill in the ocean).

Typical strategies involve either resampling (reducing the majority class, enlarging the minority class, or both) or compensating for this difference by assigning a higher penalization when the model fails to predict the minority class, making these failures more “expensive”.

3.7.2 Random undersampling

The term undersampling refers to the treatment of the majority class.

This technique consists of using just part of the observations in the majority class, chosen randomly, and all observations from the minority class so that the samples for each class have approximately the same dimension.

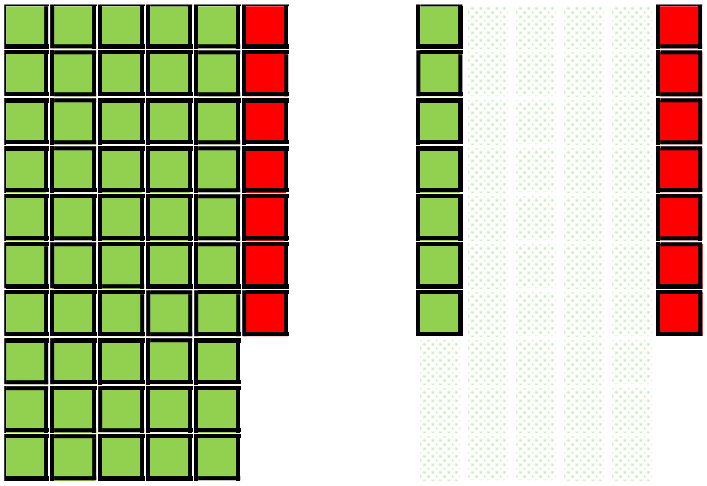

Figure 3.6 - Undersampling of an imbalanced dataset

3.7.3 Random oversampling

The term oversampling refers to the treatment of the minority class.

It takes all observations from both classes, then extracts repeated observations from the minority class until both classes are approximately balanced.

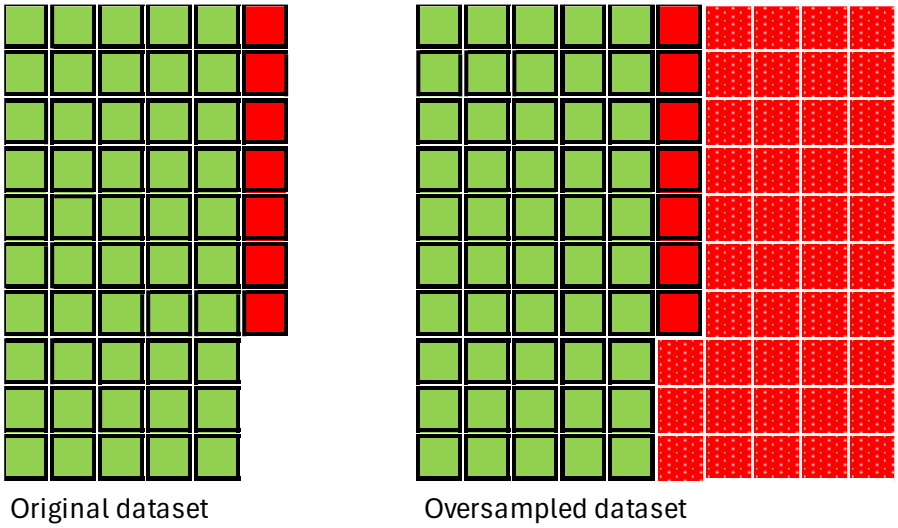

Figure 3.7 - Oversampling of an imbalanced dataset

3.7.4 Synthetic Minority Oversampling TEchnique (SMOTE)

SMOTE is a variation of Random Oversampling.

All observations from both the majority and minority class of the train dataset are selected. Then, an observation from the minority class is randomly selected, and a number (usually, two) of its nearest neighbors of the minority class are identified. A new synthetic observation is created by interpolation (Chawla et al., 2002), between the originally-selected observation and each of these neighbors.

3.7.5 Adaptive Synthetic sampling (ADASYN)

ADASYN is also an oversampling technique. The difference is that, instead of selecting random observations from the minority class for generating new observations, a weight is assigned so that the instances that were harder to learn are more likely to be represented than those that were easier to learn (He et al., 2008).

This approach, while compensating the issue of data imbalance, also shifts the decision frontier towards the area where instances are harder to predict.

3.8 MODELS

3.8.1 Motivation

The model is responsible for making the predictions. All preprocessing performed can impact the predictive capability of the system, but this can only be measured by using a model.

Different models will have different strengths and weaknesses, most of which result from the way in which data is analyzed and organized. It is therefore of paramount importance to understand how each specific model works.

The first model presented, the Decision Tree, is in fact also the building block for the following three models - Random Forest, XGBoost and LightGBM. These are the models tested to make predictions on fraud. The two main reasons for that are that tree-based models present top performance with tabular data (Grinsztajn et al., 2022) and that they provide better explainability than most other models. The description that follows, for these models, will be for binary classification problems, unless otherwise mentioned.

Lastly, a multilinear regression model will be used in what is a regression problem. In this case the requirement is no longer to predict results, but to determine the relative influence of each option on the overall result, and whether or not that influence is statistically significant.

3.8.2 Decision Tree

A Decision Tree is a representation of a function that receives an observation and returns a decision (Russell et al., 2022, p. 675). This decision is produced by conducting a sequence of

tests that start in the root and end when a leaf is reached. At each node (split), the condition is assessed, and one of the paths is followed until a leaf is reached. The leaf has a prediction associated with it, which will be the prediction returned for that observation.

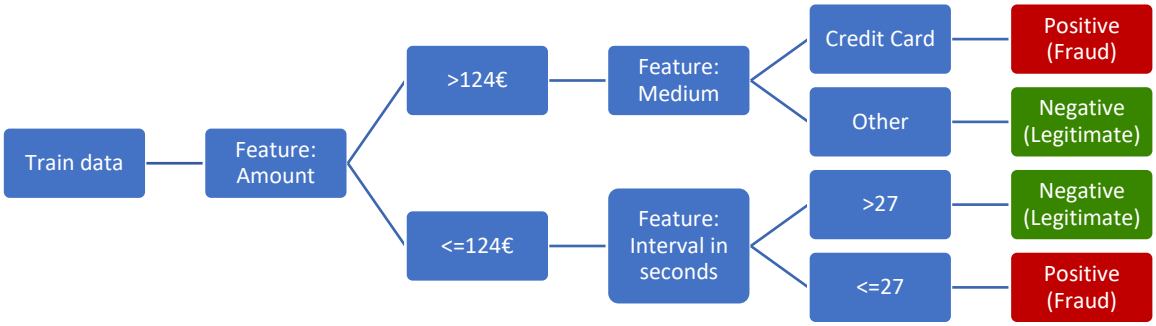

Figure 3.8 - Representation of a Decision Tree

After the Decision Tree is built, each prediction can be explained by the conjunction of all expressions evaluated. A credit card payment of 200 euros will be predicted as fraud because the amount is above 124 euros and the medium used was a credit card (top path in the image).

In order to build the tree, each value for each feature available is analyzed to determine the pair (feature and value) that allows for a better separation of the classes, therefore maximizing information gain (reducing entropy). This becomes the first node and the first two leaves. The train dataset is then divided in two, based on this condition. If a leaf has instances of only one class, it is said to be pure. For leaves that are not pure, a split will be decided using the same process, now calculated only on the corresponding subset of the train data. This leaf will then become a branch with a split condition that defines two new branches that lead to two new leaves, with the dataset being further subdivided. The process is repeated recursively until all leaves are pure or until a defined stop condition is met (e.g., maximum tree levels, minimum samples per leaf).

In the end, each leaf's composition determines its prediction.

When evaluating the best combination of features and split thresholds, one criterion is choosing the pair that provides the highest information gain. Information Gain (IG) is the reduction in entropy obtained by splitting compared to the initial entropy (without split). Another criterion that can be used for splitting is the Gini Index (Raileanu & Stoffel, 2004).

This approach is greedy, in the sense that the optimization will take the decision that provides the best immediate result, regardless of the impact it may have on the splits that will follow. In practice, the best split at one moment may result in the following splits being not so good, while a different initial split could have more promising results down the line. There is no mechanism for going back and reevaluating splits.

Decision Trees tend to overfit to the train data, especially if left to grow unrestricted. Several techniques exist (e.g., pruning) that can reduce this problem.

Decision Trees provide high interpretability. It is possible to describe why a certain observation was predicted to be of a certain class, using a set of conditions stated in plain English.

3.8.3 Random Forest

Random Forest is a model that takes a number of Decision Trees and trains each of them independently. This would lead to the creation of a lot of similar trees, as the best split would always be on the same feature and value (Russell et al., 2022, p. 715). Random Forests address this by randomly selecting the features that are made available for the model at each split. By ensuring variability between Decision Trees in a Random Forest, we are also reducing the Decision Tree's natural tendency to overfit, when left to grow uncontrolled.

A Random Forest makes predictions by combining the prediction of each Decision Tree.

Random Forests frequently present a very high performance, while still maintaining some explainability.

3.8.4 Extreme Gradient Boosting (XGBoost)

Boosting is a technique that involves sequentially trained instances of a model. Each instance receives information on what observations were correctly and incorrectly classified by the preceding instance. Boosting ensures that the model will be more "attentive" to these observations than to the rest, by increasing the weight of misclassified observations and decreasing it for the rest. This means that each instance will attempt to lessen the error from the previous one.

In gradient boosting, additional attention is given to the gradient between correctly and incorrectly predicted observations.

Extreme gradient boosting (XGBoost) is an implementation of Gradient Boosting that uses Decision Trees as the base model (Sultana et al., 2023). It introduces pruning, with a concern for efficiency, and a regularization term, to penalize more complex trees and therefore reduce overfitting.

It was implemented so that it is parallelizable across multiple machines (Russell et al., 2022, p. 719). XGBoost is a very popular model in both the industry and in Machine Learning competitions.

3.8.5 Light Gradient Boosting machine (LightGBM)

LightGBM is an algorithm that is popular in fraud detection (A. Aslam & Hussain, 2024). It was created by Microsoft to improve on memory usage and performance of XGBoost. It addresses

the computational bottleneck in XGBoost, that requires the evaluation of all possible splits and all data instances, in order to find the maximum possible information gain.

LightGBM (Ke et al., 2017) makes two simplifications that significantly reduce computational cost, while still providing a very good approximation to the ideal splitting point.

The first simplification is using Gradient-based One-Sided Sampling (GOSS). It excludes a significant proportion of data instances with small gradients and uses only the rest to estimate the information gain. By evaluating only the data instances with larger gradients, which have significantly more impact on information gain, LightGBM still obtains a very accurate estimation, but using a lot less computational resources.

The second simplification is using Exclusive Feature Bundling (EFB). By bundling mutually exclusive features (i.e., features that rarely take nonzero values simultaneously), the number of features can be reduced. Calculating the optimal bundling of exclusive features using a greedy algorithm provides a very good approximation, so that the accuracy of split point determination is still high.

While these simplifications have small or no impact on determining the split points, they significantly reduce computational costs when compared to XGBoost.

3.8.6 Linear Regression/Multilinear regression

All models mentioned previously are (or can be used as) classification models, meaning that for a given observation, these models will predict its class. Classes are discrete values and can, as is the case of fraud, be binary: positive (fraud) and negative (legitimate).

Linear regression is a regression model. It predicts a continuous quantity, not a class.

A Linear Regression takes a group of k features, the explanatory variables $(X_1, X_2, \dots X_k)$ and assigns weights $(\beta_1, \beta_2, \dots \beta_k)$ to each of these variables, plus a weight that is independent of the variables (β_0).

In order to predict the value of the dependent variable, Y, the value of each feature is multiplied by the corresponding weight, and the intercept is added:

$$Y = \beta_0 \ + \ \beta_1 X_1 \ + \ \ + \dots + \beta_k X_k$$

In general, this prediction will not coincide exactly with the observed value. Mathematically we improve the model's predictive capabilities by minimizing residuals, such as the Root Mean Square Error (RMSE) or the Mean Absolute Error (MAE).

Besides the weight assigned to each variable, the linear regression also returns a "level of confidence", the p-value, to each of the features in the dataset.

Regularization can be employed to reduce train time and to reduce overfitting. Both L1 and L2 regularization improve results (Ng, 2004), but L1 has better results.

3.9 METRICS

3.9.1 Motivation

A metric is a criteria used to evaluate the performance of a model. For a binary classification problem, some simple metrics answer questions such as: “of all predictions, how many times was the model correct?” or “of all positive cases, what percentage was correctly predicted by the model?”.

Comparing different models’ performance requires measuring the results they achieve under similar circumstances. There is no one-size-fits-all measurement in Machine Learning. Each metric has merits and disadvantages; understanding those is key for choosing the most adequate for the problem.

For convenience and simplicity, the explanation provided will be for binary classification problems. We have 4 measures of special interest; True Positives (TP) and True Negatives (TN) are the transactions for which the model correctly identified the class, as either fraud (TP) or legitimate (TN). False Positives (FP) are transactions identified fraud, when in fact they were legitimate. False Negatives (FN) are frauds that were not identified, because the model considered them to be legitimate transactions.

3.9.2 Accuracy, Precision, Recall or True Positive rate, True Negative rate

Accuracy answers the question: “What is the ratio of correct predictions”?

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

This metric has the advantage that it considers all predictions and is very easy to interpret. However, for highly imbalanced datasets the results may be counterintuitive. If we have only one fraud per thousand transactions, by considering that all transactions are legitimate, the model will not catch any fraud, but its accuracy will be 999/100, or 99.9%. The accuracy in this case is not measuring the capacity for detecting fraud, but the imbalance of the dataset.

Precision answers the question: “When the model predicts a positive, what’s the ratio of those predictions that are correct”?

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high value means that when the model predicts fraud, it is indeed likely that the transaction is fraudulent. This ratio is calculated on the cases predicted to be fraud; it will decrease if the model incorrectly predicts fraud (FP), but will not decrease if a fraud is allowed (FN is not part of the equation). So, optimizing for precision means that if a model is uncertain of whether a transaction is a fraud or not, it will prefer to allow it, as there’s no penalty for FN.

Recall answers the question: “Of all the fraud attempts, what’s the ratio that the model detected”?

$$\text{Recall} = \frac{TP}{TP + FN} \;=\; \text{True Positive Rate} \;=\; \text{Sensitivity}$$

In this case, Recall is penalized if a fraud is allowed (FN), but not if a legitimate transaction is refused (FP). Considering that the cost of accepting a fraud is higher than the cost of rejecting a legitimate transaction, it can be argued that Recall is a more relevant measure than Precision. For both metrics, it can be said that they can be easily tricked by a model that always predicts Fraud or Legitimate.

Recall is also called the True Positive rate or Sensitivity.

The True Negative rate, or Specificity, measures the number of negatives that were correctly predicted:

$$\text{True Negative Rate} = \frac{TN}{TN + FP} \;=\; \text{Specificity}$$

This measures the ratio of legitimate transactions that is correctly predicted not to be fraud.

3.9.3 Area under the ROC curve (AUROC)

The Receiver Operating Characteristics (ROC) curve compares the results for Sensitivity and Specificity, or the True Positive Rate and True Negative Rate. The curve represents the balance between TP rate and TN rate at different thresholds, by drawing a diagonal we get the result expected through random choice, the further the ROC curve is above that line, the better the model.

The numeric reading of this curve is the area under the curve, which in practice will be between 0.5 (the diagonal) and 1.0 (a model that is perfect in predicting at all thresholds.

3.9.4 F1 Score

F1 score tries to attain a balance between Precision and Recall. In fact, F1 score is the harmonic mean of Precision and Recall:

$$\text{F1Score} = 2^{*}\left(\frac{Precision * Recall}{Precision + Recall}\right) = \frac{2 * TP}{2 * TP + FP + FN}$$

With this metric, if the model is optimizing for Precision, the result will be penalized through a lower Recall, and vice-versa. It becomes impossible to attain a great result by using simplistic approaches such as predicting always the class that benefits the metric.

For binary classification problems, the F1 score only performs calculations on the positive class. Other options exist, in which the F1 score is calculated separately for each class and is then combined. Of more interest are the “F1 score macro”, which returns the mean of the F1

scores for each class, and the “F1 score weighted”, which calculates a weighted mean. Comparatively, the “F1 score macro” will assign more relevance to the minority class, which can make this class significantly more influential than the representation it has on the dataset.

3.9.5 Custom metrics

One approach that has been explored (Hand et al., 2008), establishes that the cost of a false negative is equal to k times the cost of a false positive. Using this assertion, optimization can then take into account both false positives and false negatives, in terms that are relevant to the business. It is of major importance that the value established for k reflects the business need.

This approach nevertheless considers all false negatives to have the exact same cost, without taking into account the amount of the transaction, which evidently is not correct. The same can be argued about false positives, in which the acquirer loses transaction fees that depend on the amount of the transaction.

A metric can be engineered to represent the actual cost of a false positive and the actual cost of a false negative, taking into account such aspects as the amount of the transaction, transaction fees, estimated reputational costs, and possibly other elements.

4 METHODOLOGY

4.1 CRISP-DM AS A FOUNDATION

The industry-standard CRISP-DM model (Wirth & Hipp, 2000) is at the foundation of the research conducted in this thesis.

CRISP-DM is very popular in the literature and is used in several industries to address different sorts of problems. In fact, it has even originated variations to address specific needs in Data Mining and Machine Learning, such as CRISP-EM for Evidence Mining (Venter et al., 2007), CRISP-TDM for Temporal Data Mining (Catley et al., 2009), CRISP-DM0 for null-hypothesis driven confirmatory Data Mining (Heath & Mcgregor, 2010), CRISP-IM for Idea Mining (Ayele, 2020), CRISP-DM(Q) for Quality Assurance (Studer et al., 2021), CRISP-eSNeP (Electronic Social Network Platforms).

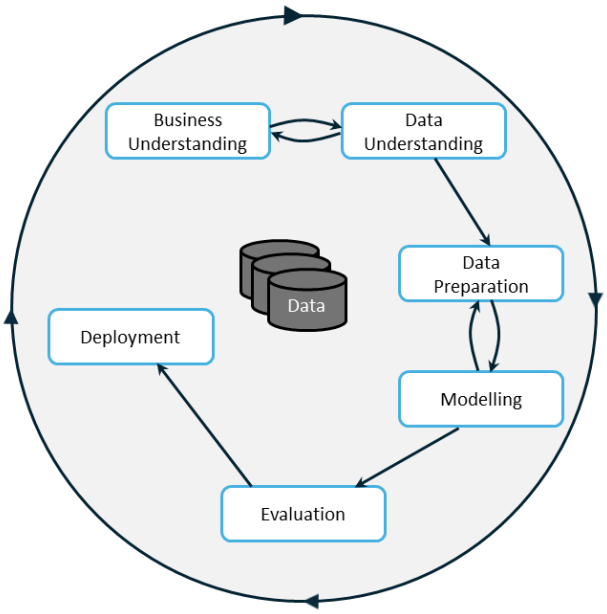

Figure 4.1 - CRISP-DM Phases and process flow (Wirth & Hipp, 2000)

CRISP-DM establishes a consistent path for a Data Mining or Machine Learning project, organized in six phases, from understanding business objectives to the deployment in a production environment of an optimized model. This path however does not have to be linear, as CRISP-DM allows for the project to move back and forward between phases.

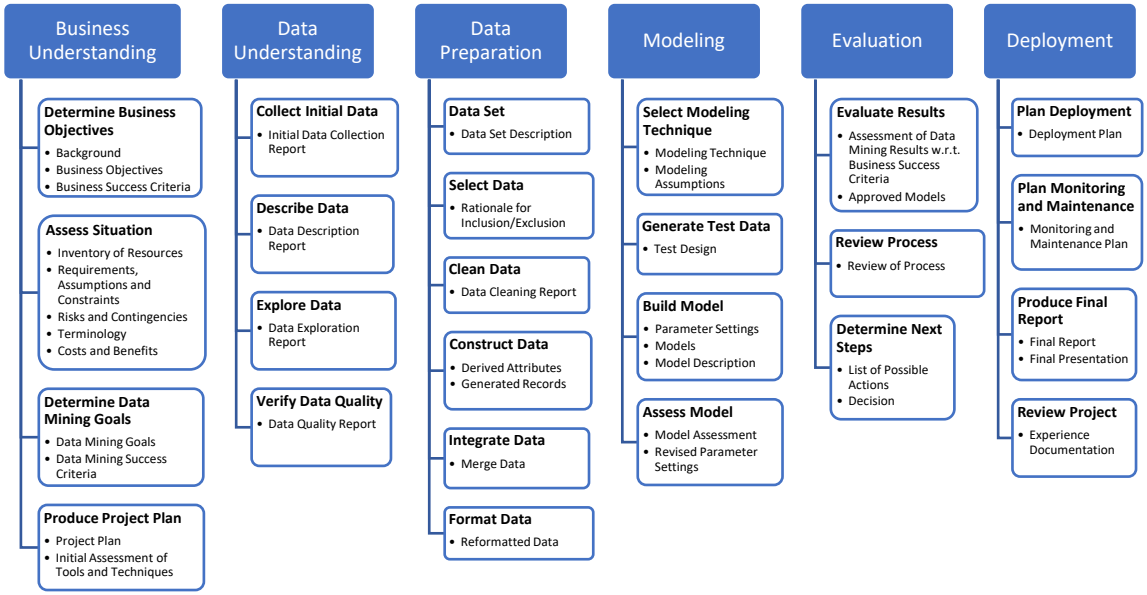

Figure 4.2 - CRISP-DM phases and their Generic Tasks (Wirth & Hipp, 2000)

This thesis analyses the decisions taken during the intermediate phases of CRISP-DM – Data understanding, Data preparation, Modeling and Evaluation. The first phase of business understanding and the last phase, deployment, albeit extremely relevant in a Machine Learning project, will contribute to the discussion of metrics and cost of false positives and false negatives, but are not central for this research.

4.2 CRISP-DM Phases and Tasks

4.2.1 Phase: Business Understanding

If a single dataset and problem is analyzed, it is possible that the Machine Learning process will be biased, and that the conclusions are valid for that reality, but not for others. Therefore, this thesis will repeat the same analysis on different datasets.

4.2.2 Phase: Data Understanding

The data understanding phase will concentrate on determining the quality of the dataset, meaning that a dataset found to be inadequate will be not further analyzed.

4.2.3 Phase: Data preparation

Data preparation is a key part of a typical Machine Learning process. By using several datasets, this task becomes more critical, as it is expected that data preparation will standardize the datasets so that at the next phase, different models can be tried on each dataset.

It is well-established that the datasets for fraud are imbalanced (Baesens, Höppner, Ortner, et al., 2021; Makki et al., 2019; Zhu et al., 2020); handling this imbalance will be experimented

using different undersampling and oversampling techniques. Even though many Machine Learning models encompass the capability of handling class imbalance, this would create an implementation dependency on the specific Machine Learning model, and that is not desirable. All models should be experimented in the same conditions.

As part of data preparation, data cleanup tasks such as inconsistencies handling, Null value handling, outlier handling and scaling will be performed.

Data engineering techniques will be employed to create synthetic features.

Finally, automatic feature selection will be conducted on each dataset, to determine which features contribute to the predicting capabilities of the models.

4.2.4 Phase: Modeling

The models analyzed will include some standard models for supervised learning, such as Decision Trees and Random Forests, as well as some models that are usually employed in fraud detection scenarios, for concerns of either model performance and/or explainability, such as XGBoost and LightGBM.

4.2.5 Phase: Evaluation

Defining how to measure models' performance is key to the research, as decisions and recommendations will be made based on the impact they have on performance.

In business terms this has some complexity, as the actual cost of a misclassification depends on whether the misclassification was a false negative (an actual fraud was allowed) or a false negative (a legitimate transaction was rejected), on the amount of the transaction, on the commissions lost, or on the (not easily measurable) reputational cost. Furthermore, depending on the situation, the cost may be incurred by the cardholder, by the merchant, by the bank, or by an insurance company.

For the purpose of this thesis, and knowing that fraud datasets are extremely imbalanced, with positive records being a very small part of the occurrences, accuracy is an inadequate measure, lest we fall into the accuracy paradox (Valverde-Albacete & Peláez-Moreno, 2014): if the percentage of positive records is extremely small, a model that always predicts the negative class with fail to identify any of the positive occurrences, but will nevertheless score extremely well in accuracy.

Other measures should be considered. Precision will measure the percentage of times that our model was correct when it predicted a positive result. Recall will measure the percentage of actual fraud that was detected by our model.

Given that a false negative (a fraud that was not detected) is more onerous than a false positive (a legitimate transaction that was refused), Recall is arguably more relevant than Precision.

Even if less impactful in economic terms, false negatives have costs that should not be disregarded, such as the loss of commissions and potential reputational cost for the bank.

One approach that has been explored (Hand et al., 2008), establishes that the cost of a false negative is equal to k times the cost of a false positive. Using this assertion, optimization can then take into account both false positives and false negatives in terms that are relevant to the business. There is, however, an arbitrary decision in establishing this correspondence.

A more standard alternative is F1-Score, the harmonic mean of Precision and Recall. This metric captures the cost of both False Positives and False Negatives:

For the reasons presented, the metric considered for this thesis is the F1 Score.

4.2.6 Phase: Deployment

This phase occurs after model performance has been established and is therefore not relevant for the purpose of this thesis.

4.3 CRISP-DM AS A METHODOLOGICAL GUIDELINE FOR THE EXPERIMENTAL STUDY

In CRISP-DM, each phase in the Reference Model encompasses several "generic tasks", which in turn are established on top of "specialized tasks".

These specialized tasks are already context-dependent, as they are situated closer to the project implementation. It is at this level that this thesis establishes the alternatives that will be assessed.

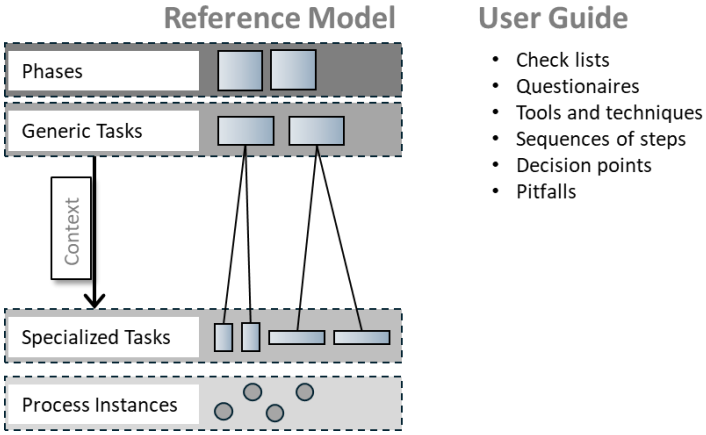

Figure 4.3 - Specialized tasks in Crisp-DM (Wirth & Hipp, 2000)

As an example, the "Data Preparation" phase has a generic task for "Clean Data". This task is composed of several specialized tasks, such as outlier handling and null value handling. For each of these specialized tasks, several approaches can be employed, and the data scientist is free to choose, for each task, the approach that will be followed in the project. Naturally, a combination of different choices will lead to different end results.

It is noteworthy that for some specialized tasks it is possible to opt for the *status quo*, meaning that no changes will be performed on the data. It is possible, for instance, to bypass outlier handling (outliers are kept and remain unchanged in the dataset) and to bypass feature engineering (no new attribute will be derived from existing attributes). This is not possible for other specialized tasks. Two obvious cases are the choice of a dataset (it is not possible to start working on data without a dataset) and the choice of a model (some model needs to be used in order to obtain predictions.

The decisions that are relevant for analysis are those taken at this atomic level. This research aims to verify the impact on the overall performance of the model of individual decisions made at this elementary level. Each combination of the individual decisions establishes the test conditions, by creating a unique ML pipeline that can be seen as a complete Machine Learning model, in the sense that it starts in the data ingestion and ends with a trained and optimized supervised model that is capable of making predictions.

At each experiment, all relevant variables and decisions that are needed to reproduce the results will be stored along with the results, so that this data can later be analyzed.

In the end, what is relevant for analysis is not the overall performance of a certain pipeline compared to the others, but the contribution of each decision for that result. The aim is to identify options that are empirically shown to be consistently better.

4.4 Degrees of freedom (DOF) of the Machine Learning project

Regardless of the fact that CRISP-DM favors iterations between phases, in the end, it is possible to envision a data processing pipeline that starts with data and ends with predictions.

The moments at which options are available constitute the "Degrees of freedom" (DOF) of the supervised model. Different options at each stage will create Machine Learning pipelines that will have similarities but also differences.

The pipeline analyzed is created with nine DOF, labeled from A to I.

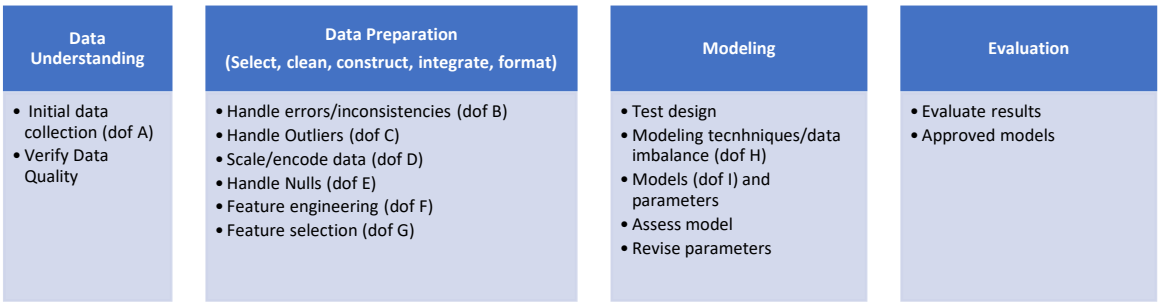

Figure 4.4 - The Machine Learning pipeline with its Degrees of Freedom

4.4.1 DOF A –Dataset

In CRISP-DM, the initial phase of Business Understanding is followed by and closely intertwines with the phase of Data understanding. The first task of Data understanding is to "Collect Initial Data".

For the purpose of this research, it is helpful to have several distinct datasets with some overall similarities (information about fraud; binary label of fraud: Yes/No), but some significant differences, such as diverse origins (mobile payments, credit card payments), nature of the data (real-world data or synthetic data), number and characteristics of the features available in the dataset (encoded; unencoded; obtained through transformations, numerical, categorical).

5 datasets were selected.

4.4.2 DOF B – Error correction

An error is a datum that is not correct.

This may be detected because it is not compatible with reality (the year of an operation is in the future; a non-existent currency is recorded for a certain transaction), is outside the scope of the dataset (in a dataset with transactions for one civil year, transactions that report another year are errors) or is inconsistent with other data in the dataset (a transaction was received before it was sent). Other cases, while constituting an error, might prove difficult or impossible to detect.

It is highly undesirable that an error is kept in the dataset; distinct approaches can be used to handle these cases, as will be presented in the experimental settings.

Error detection is not a task that can easily be automated; even simple cases require a good understanding of the data and possibly of the ways in which it was compiled. This results in the need for a manual, deep analysis of the dataset, defining what are the error conditions.

2 different approaches for error correction were chosen.

4.4.3 DOF C – Handling outliers

An outlier is a value that is credible but nevertheless differs significantly from other values in the dataset. This could be an amount that is significantly higher or lower than expected. For categorical values, the concept is associated with a categorical value that has a very low frequency. The analysis performed will be focused on the numerical features present in the dataset.

Errors and outliers are different in several aspects; a person performing a transaction 100 years before being born is certainly an error (on the age, date of transaction or both); a person

performing a transaction 100 years after being born is likely to be an outlier, nevertheless it's not something that can lightly be regarded as an error, given that it is indeed possible.

Outliers can be detected and handled using statistical or mathematical analysis. It is therefore simpler to automate the detection and handling of outliers than it is of errors.

4 distinct techniques were created for outlier handling.

4.4.4 DOF D – Scaling

Features such as transaction amount might vary from near zero to large values, others such as number of credit cards owned will have a much smaller amplitude, and yet others (such as binary features) will only vary between zero and one.

Some models (e.g., KNN) rely on the measurement of distances between datapoints and will be extremely sensitive to features that are on a much larger scale; others (e.g., Artificial Neural Networks) perform mathematical operations on values of the features and are therefore also sensitive to scale, while some others (e.g. Decision Trees) are less dependent, as the actual scale of the values is not relevant, because values are used for deciding on the best splits, but the loss function is based on measures such as Information Gain and Gini index, that do not depend on the actual values of the features.

Scaling consists in ensuring that distinct features in the dataset are presented in similar scales, meaning that they vary in the same interval.

4 different approaches were implemented for scaling.

4.4.5 DOF E – Null-value handling

The absence of value in a feature will be represented by the presence of a Null value. Some of these Null values are part of the dataset, yet some others might result from one of the approaches followed for error correction: replace errors with a Null.

Null values create difficulties for most algorithms, as they are difficult to incorporate in mathematical and statistical operations such as calculating distances or averages.

Common strategies for removing null values are apparently inspired by two of the Machine Learning tribes, the Bayesians, and the Analogizers: one common option is to replace them with a calculated metric such as mean (for metric features) and mode (for categorical features). Another common strategy is to determine the k nearest neighbors and fill the missing feature based on the values of those neighbors.

2 techniques were implemented for null value handling.

4.4.6 DOF F – Feature Engineering

Feature engineering is the creation of new features in the dataset.

Features may be created by processes that analyze the dataset and automatically build new features, as is the case of Principal Component Analysis (PCA) and the case of Linear Discriminant Analysis (LDA).

Features can also be created by experts who determine combinations of features that may yield better results than the individual use of the combined features.

4.4.7 DOF G – Feature Selection

When using a dataset with a large number of features, in what is mentioned as "the curse of dimensionality"(Verleysen & François, 2005), the space becomes more sparse, because pairs of observations are unlikely to be very similar simultaneously on all dimensions.

By selecting the features that show the most promising results (Guyon & Elisseeff, 2003), the models' effort will not be lost on features that do not seem to contribute significantly to its predictive capacity.

Two techniques that are fully automated were employed, SelectKBest and Recursive Feature Elimination (RFE), along with a customized function that performs several statistic tests to try to determine what features contribute, and what features do not.

4.4.8 DOF H – Data Imbalance adjustment

Fraud detection dataset are highly imbalanced, with a very small proportion of positive (fraud) cases. This creates a bias towards the majority class and may imply that models fail to capture the structure of the minority class.

The reduction of the imbalance may only be performed on train data, not on actual validation and test data. Two base techniques are undersampling, in which all observations are selected from the minority class, but a reduced number of samples is chosen from the majority class, and oversampling, in which all observations from the majority class are selected, and on the minority class additional observations are considered (either by repeating or fabricating new observations).

4.4.9 DOF I – Models

The actual predictions are performed using supervised models. It is at this stage that it becomes possible to evaluate how successful the pipeline is in predicting the labels for unseen observations.

In order to make these predictions, a supervised model starts by using a set of data for which a label is known (called the train data) and attempts to discover the patterns and transformations that lead from the data in a certain record to the corresponding label.

If indeed these patterns and transformations are captured by the model, it will be capable of correctly predicting the label for data that was not made available during training (this set of data is called the validation data). This capacity to predict labels on new data is what the data scientist is trying to achieve; it is said that the model generalizes well.

If the model fails to determine these rules even in the train set, it is said to be underfitting, meaning that it is not adapting as expected. The model needs to be fine tuned so that it becomes more capable of capturing the patterns that will allow for better predictions. Several conditions may contribute to underfitting, including the model being too simple or not enough features being available.

It may also occur that the model is very good on the train data but then gets poor results on the validation data. In this case, it is said to be overfitting, and this means that it is simply "memorizing" the train data, and not capturing the phenomena that explain why each observation belongs to each class. Again, several conditions may cause overfitting, including not having enough train data and the model being excessively complex. Some models, as is the case of Decision Trees, are known to be prone to overfitting.

4.5 CROSS-VALIDATION

4.5.1 The holdout method

The holdout method allows for quick results. A random split is performed on the data, dividing it into train and validation (optionally, also test). The model is trained on the train data, and then evaluated on both train and validation data.

After the first execution, a result is immediately available. The drawback is that this result may be significantly influenced by the split; if, for example, a few outliers exist, and all of them happen to be on the validation set, this will influence validation performance. For this reason, the holdout method's results are partly explained by the model's performance, but potentially also by the split that was performed, and typically we cannot remove this bias.

By comparing results on the train and validation datasets, we can detect underfitting and overfitting and change the model's structure or hyperparameters, in order to improve it.

The test dataset, which is reserved for evaluating a model after it is fine-tuned, will not be used for metrics. It is only used to assess the performance of the optimized model, which is outside the scope of this research.

4.5.2 K-fold Repeated Cross-validation

K-fold Cross-validation, or simply cross validation, is a commonly-employed technique that combines train and validation data and performs a pre-defined number of splits (K). Then, K iterations are run, training the model on all combined splits except one, and validating the trained model on that split that was left out. After K runs, each record was present once in the

validation data set, and K-1 times in the train set. The cross-validation performance is obtained by averaging the results of each run.

Iteration 1	**VALIDATION**	Train	Train	Train	Train
Iteration 2	Train	**VALIDATION**	Train	Train	Train
Iteration 3	Train	Train	**VALIDATION**	Train	Train
Iteration 4	Train	Train	Train	**VALIDATION**	Train
Iteration 5	Train	Train	Train	Train	**VALIDATION**

Figure 4.5 - Cross-validation diagram

In repeated cross-validation with n repetitions, after splitting data in K folds and evaluating performance as described, new splits are performed, and new runs are executed, in a total of n cycles. In the end, the measures will be the average performance of K splits x n runs.

Repeated cross-validation, with 10 folds and 3 repetitions, results in a total of 30 runs. This ensures that the results not only are free from any relevant split bias but have statistical significance. It needs to be acknowledged that, however, cross-validation and repeated cross-validation take significantly more time to evaluate a model than the holdout method.

In fact, using K-fold 10 means that the model will be trained 10 times on the train data and each of these trains will then generate predictions on the validation data. This means that the time needed will be similar to performing 10 times the holdout method. Or 30 times, if we consider repeated cross-validation with a repetition factor of 3.

Finally, it is worth noting that large datasets are less susceptible to the bias that may be introduced by the split between train and validation and, therefore, it is in smaller datasets that cross-validation is of paramount importance.

4.6 EVALUATING RESULTS

4.6.1 Methodology for measuring results

Knowing that the different options will have interactions, results will be measured using a complete pipeline, then the decisions at each “DOF” will be recorded, and the results for different metrics will also be collected.

As mentioned before, the pipelines have 9 DOF, meaning that there are 9 moments in which a selection has to be made, from a list of possible options. These options will be presented in detail later, but the following table summarizes the number of options that was implemented for each DOF:

Table 4.1 - Pipeline options, by DOF

DOF		# options
A	Dataset selection	5
B	Error handling	2
C	Outlier handling	4
D	Scaling	4
E	Nulls	2
F	Feature engineering	4
G	Feature selection	4
H	Resampling	3
I	Model	4

4.6.2 Measuring results – Brute force approach

In order to find the absolute best pipeline, it is necessary to perform all combinations of options, measure the results for all of them and choose the one that scores highest.

The number of unique pipelines, meaning that at least one option is different from all the other pipelines, is obtained by multiplying the number of options at each step (DOF):

5 * 2 * 4 * 4 * 2 * 4 * 4 * 3 * 4 = 61.440.

In order to have statistical significance, each pipeline has to be trained and evaluated at least 30 times, with different datasets. So the actual number of runs needed is:

61.440 * 30 = 1.843.200

Using a reasonable but limited processing capacity, it would be a challenge to determine the performance of such a large number of pipelines. This option is, therefore, not realistic and is discarded. In order for this approach to be feasible, it would be necessary to both reduce the number of DOF and the options available at each DOF.

Moreover, the direct analysis of the results at the end of the pipeline would fail to provide direct explanations regarding the individual contribution of the options at each stage along the process and would potentially put the emphasis on the best combination of options, not on the best option at each step of the way. While this may make sense for training one specific model on a specific dataset, it could potentially lead to conclusions that depend on the dataset and, therefore, are less likely to be valid in conditions that differ from the test setup.

4.6.3 Measuring results – *Ceteris Paribus* approach

In economics, *ceteris paribus* (translated as “keeping all other options fixed” or “all other things being equal”) is used to study the impact on a certain metric of changing one

independent variable, while keeping all other variables constant. E.g., impact on demand of an increase in a product's price, while the level of publicity remains unchanged.

Translated to the evaluation of an ML pipeline, this approach evaluates the impact on the pipeline's result after fixing the configurations for all steps except one, then for that step experiment with the different options. The best result will show what's the best option at that stage. Then, this option is fixed, and the experimentation is repeated by changing just one other step, and so on until all steps have been checked.

In this approach, the number of configurations for each DOF is added, not multiplied, because when we're experimenting the options in DOF_A, all others are kept constant. The number is now perfectly reasonable:

5 + 2 + 4 + 4 + 2 + 4 + 4 + 3 + 4 = 32.

Each configuration needs to be run 30 times:

32 * 30 = 960

While this number is easy to attain, careful attention needs to be paid to this approach. It is somewhat naïve to disregard the interaction between options; certain options might be more significant depending on other options. By fixing all the steps along the pipeline, it will not be possible to capture these interactions, significantly influencing the results obtained.

In real-world circumstances, this type of *ceteris paribus* conditions are not common to find and not easy to implement.

4.7 EVALUATING RESULTS - A NOVEL APPROACH LEADING TO A NOVEL METHODOLOGY

Mater artium necessitas

(necessity is the mother of invention)

Knowing that the "brute force" approach is unrealistic as it would demand executing millions of runs, and the "ceteris paribus" is naïve in ignoring interactions between options, this study proposes a novel approach.

4.7.1 Grafting: Brute Force + *Ceteris Paribus*

It would be ideal to attain the benefits of the brute force approach, such as getting information for all possible configurations, while also getting the benefits of the *Ceteris Paribus* approach, specially the possibility of obtaining results from a reduced number of runs. In many languages, popular culture says that it is not possible to have the best of both worlds. Yet, the human drive to move forward is a powerful force; and ingenuity creates solutions such as grafting, that is used since ancient Rome to improve agricultural production.

The approach proposed is loosely inspired by grafting in the sense that it tries to combine the benefits of both approaches, but in truth, it does not implement one after the other.

The tests will be conducted using the brute force approach, in the sense that each option in one DOF may be tested with any other option in all other DOFs. However, instead of systematically creating all possible pipelines, in each run a new pipeline will be assembled by randomly choosing one option for each DOF. The tests will not be exhaustive; DOF_A_1 may be selected in many runs, but will not be combined with all variations for all other DOFs.

In fact, given that the number of tests will be significantly lower than the number of possible pipelines, it is certain that the majority of possible pipelines will never be executed.

Even by running thousands of randomly assembled pipelines, there is a small (but never zero) probability that some option will never be chosen, or is chosen a very small number of times. Post-hoc control will be exerted, to make sure that each configuration gets significant coverage.

By allowing for all DOF to vary at the same time, there is no assumption that these options are independent, as would be necessary for the "ceteris paribus" approach. In fact, it is desirable that many different combinations are evaluated so that the interactions between different configurations will be allowed to influence the results, as is desirable.

4.7.2 Holdout method: a superior choice!

By using repeated k-fold cross validation, each pipeline is executed 30 times, with the exact same configuration, significantly reducing the risks associated with a single split of the dataset. However, as time is a restriction, there are 29 different pipelines that could be experimented with, with different configurations. This is a better option, and in fact the benefit of reducing bias is better achieved by using more configurations, rather than improving the assessment for a specific configuration by running it 30 times. This means that the holdout method is in fact a better option, and so it will be employed. This maximizes the distinct configurations that will be run, without the concern that a specific configuration must get a perfect evaluation.

Using the results obtained for a large number of diverse pipelines, a multilinear regression (another model may be used, provided that it has good interpretability) will be employed to analyze the pipelines' results. Options for each DOF will be one-hot encoded, thereby creating explanatory variables. This approach not only calculates the contribution of each option for the results, but also provides a p-value to attest the statistical significance of that conclusion.

This is still a demanding approach, as a large enough number of runs has to be conducted, but it offers several benefits over both the "brute force" and the "ceteris paribus" approaches.

4.7.3 Building the multilinear regression

After the tests are conducted as described, and the configurations and metrics are logged, the next step is to explain each metric based on the configuration.

The Linear Regression model will be informed of what is the feature that we want to explain (the dependent variable, for instance, the F1 Score) and informed of what features are available to explain it (the independent variables or explanatory variables). The explanatory variables correspond to the one-hot encoding of the options available at each DOF.

As long as several metrics are collected during the runs, it is always possible to create a new Linear Regression to explain another metric, just by changing the dependent variable.

4.7.4 Multilinear regression - baseline result

After the Linear Regression is fitted to the data, it will assign results to the variables, with one "special variable" that is the baseline for the model. The baseline is the configuration in which, at each DOF, option 1 was chosen. In practice, the baseline corresponds to this pipeline:

A_1 > B_1 > C_1 > D_1 > E_1 > F_1 > G_1 > H_1 > I_1

These variables, DOF_A_1, DOF_B_1, DOF_C_1, etc, are not present as independent variables, because they are employed in the pipeline.

The baseline result does not come from experimentation with that specific configuration - in fact, it is likely that this specific configuration was never assembled but inferred from the results of all the runs.

4.7.5 Multilinear regression - additional results

As for the rest: DOF_A_2, DOF_A_3, DOF_B_2, and so on, each will have an assigned weight and a p-value. The way to interpret the result for DOF_A_2 is that the model expects the dependent variable (our metric) to increase by that amount, if we use DOF_A_2 instead of the option in the baseline (DOF_A_1).

To estimate the result for a certain configuration, we can simply take the value for the baseline and add up the predictions for the options that are not part of the baseline.

The p-value associated represents the statistical confidence in that result. If p-value is higher than 0.05, the result may be a spurious result and should not be considered. If it is 0.05 or smaller, there is no statistical evidence to suggest discarding this result, so it should be considered as credible.

4.7.6 Multilinear regression - further analysis

Analysis such as explaining the impact on the F1 score of a certain option along the pipeline will likely come to mind and are indeed straightforward to conduct. There is, nevertheless, a lot more exploration that can be done.

Other metrics that are being recorded, such as precision, recall, train time, prediction time, as well as other metrics that can be engineered based on the logged information. The can then be defined as a new target variable, and the refitted model will be explaining that new variable.

5 EXPERIMENTAL STUDY

5.1 CONTEXT

In the typical Machine Learning supervised project, data is incrementally handled and processed, starting from data that is made available for analysis and ending in a model that is trained and capable of making predictions. At each of these incremental steps, it is common to consider more than one option. Typically, one option is chosen and the others are discarded; sometimes and in the spirit of CRISP-DM, some experimentation is conducted on which option yields the best results, but finally that option is crystalized and the process moves to the next step.

The fact that there are several alternative options at a certain step constitutes a "Degree of Freedom" (DOF), in the sense that several paths can be chosen and followed, and in the sense that a decision is required in order to move forward.

The purpose of the experimental study is to measure the impact of each individual decision when compared to alternative options.

5.2 DEGREES OF FREEDOM (DOF)

As an example, a Degree of Freedom (DOF) available is the method used for scaling numeric data. This happens to be DOF_D. Options made available are: DOF_D_01: no scaling; DOF_D_2: use MinMaxScaler; DOF_D_3: use StandardScaler; DOF_D_4: use RobustScaler.

For each run of the model, nine degrees of freedom are considered, each one with several alternative options, as detailed below.

5.2.1 DOF_A – Dataset

The datasets procured for the experimentation are a key part of this study. They provide a rich and diverse environment on top of which all techniques will be employed. Datasets were chosen based on being publicly available and relating to financial fraud, and having a binary label to signal whether a certain operation is a fraud, with fraud being the positive label. Some of the datasets employed refer to real transactions, while others are synthetic, meaning that they were fabricated to represent the reality that is found in actual data.

DOF_A_1 - Credit Card Fraud Detection

This dataset has transactions made by European credit card holders during two days in September 2013. It has 31 features, 28 of which were created using PCA to provide anonymity. Features Time and Amount and the fraud label were not obfuscated.

The dataset is publicly available (Université Libre de Bruxelles, 2018), as part of the investigation on fraud (Lebichot et al., 2021; Pozzolo, 2015).

DOF_A_2 – Bank Account Fraud (BAF)

This is a dataset created and made publicly available by Feedzai, a Portuguese company that specializes in financial fraud services. It was published at NeurIPS 2022, a conference dedicated to Neural Information Processing Systems. The data was synthetically generated, and the company claims that it is crafted to be realistic and complete so that it constitutes a robust test bed for ML experimentation. The dataset has 6 variants, out of which only the "base" variant will be used. The base variant of the dataset includes 32 features and has 1.000.000 observations.

The dataset is publicly available (Jesus, Pombal, & Saleiro, 2022) and was created as part of a publication (Jesus, Pombal, Alves, et al., 2022).

DOF_A_3 - Synthetic data from a financial payment system

This dataset is also available on Kaggle.com. It was generated using the BankSim payments simulator to generate data for 180 days of payments. Aggregated data from a bank in Spain was used as input, and parameters were controlled to ensure a distribution of data that is close enough to actual data and that can be reliably used for testing. The dataset has 594.643 observations with 10 features.

This dataset is publicly available (E. Lopez-Rojas, 2017), following the publication in which BankSim was proposed (E. A. Lopez-Rojas & Axelsson, 2014).

DOF_A_4 – Forage website, www.forage.com

Forage's website provides challenges in several areas, as a means for candidates to demonstrate their skills when applying for a job. The dataset was obtained from one of those challenges. It has 200.000 observations with 11 features.

This dataset is published as part of the theforage.com website (*Forage*, 2023).

DOF_A_5 – Synthetic Financial Datasets for fraud detection

This dataset is available on Kaggle.com. It was generated using the PaySim payments simulator with aggregated data from a multinational company that provides mobile financial services in more than 14 countries across the world and using one month of financial logs from one African country. The dataset has 6.362.620 observations and 11 features.

This dataset is publicly available (E. Lopez-Rojas, 2016) and was created as part of a publication on applying simulation for fraud detection (E. A. Lopez-Rojas, 2016).

5.2.2 DOF_B – Error correction

For each dataset, a configuration file was created with a structure that allows for the definition of several validations. When a record fails one of these validations, two approaches are defined; either the record is deleted from the train dataset, or the attribute that failed the validation is replaced with a Null value.

Record elimination is only done on the train dataset; regardless of the error correction option that was randomly chosen. In the validation and test datasets, the correction is always the assignment of a Null value.

DOF_B_1 – Replace with Null

Error values are replaced with Null on all datasets – train, validation, and test.

DOF_B_2 – Drop from train

Error values on the train dataset are dropped.

Error values on the validation and test dataset are replaced with Null.

5.2.3 DOF_C – Outlier handling

Outlier handling is part of a typical Machine Learning pipeline. Outliers, being observations that are not representative of the population, may significantly influence the data processing, by shifting the mean for a certain feature, or by making a certain observation seem very distant from all others, in scenarios where distance is computed.

Examples include the imputation of Null values with either the mean for a certain feature or using KNN. In fact, all cases where distance between points is measured, will be significantly impacted by the existence of outliers.

DOF_C_1 – Ignore outliers

This option corresponds to not handling outliers, meaning that those observations will remain unchanged in the dataset. Intuitively this seems to be a sub-optimal solution, but nevertheless it is a possible option, and one that we will want to control.

DOF_C_2 – Update value, consider IQR x 3

For each feature, the interquartile range, or IQR, corresponds to the difference between the first and third quartiles (Q1 and Q3). In this method, we will establish a lower boundary corresponding to Q1 - 3 x IQR, and any number below that will be updated to this boundary. Symmetrically, an upper boundary of Q3 + 3 x IQR will be established, and any number above that boundary will be updated to that limit.

This strategy does not avoid outliers, but simply brings them closer to the remaining observations.

DOF_C_3 – Update value, consider IQR x 1.5

This approach is exactly the same as was described in DOF_C_02, with the only difference that the boundaries defined are Q1 – 1.5 x IQR and Q3 + 1.5 IQR.

DOF_C_4 – Delete outliers (IQR x 1.5)

If this strategy is chosen, records with outlier values are removed from the train set.

Outliers in this case are defined as values below Q1 – 1.5 x IQR or above Q3 + 1.5 x IQR.

5.2.4 DOF_D - Scaling

When handling numeric values, many models and computations rely on all features being in the same scale. Several techniques were implemented to scale data.

DOF_D_1 – Do not scale data

If this is the option, no change is performed in the data.

DOF_D_2 – MinMaxScaler

Features will be scaled down (or up) to a certain interval. The interval used is [0,1] so, for each feature, the lowest value in the train dataset will be converted to 0, the highest will be converted to 1, and all values in between will be updated proportionally.

DOF_D_3 – StandardScaler

Standard Scaler takes features that follow a Normal distribution and transforms them, so that the mean will be zero and the standard deviation will be 1.

DOF_D_4 – RobustScaler

RobustScaler is similar to StandardScaler, with the difference that the distribution is based on quartiles and not on the mean and standard deviation and is, therefore, more robust to outliers.

5.2.5 DOF_E - Null Value handling

Null values in our dataset may correspond to the absence of information, or to errors that were corrected by assigning a Null. These unknown features pose a problem for many models; a strategy needs to be defined to assign them a value.

DOF_E_1 – Median and Mode

For numerical features, the median will be calculated on the train set, and that median will be used to update train, validation, and test datasets.

For categorical features, the mode will be determined on the train dataset, and will then be assigned to Null values on the train, validation, and test datasets.

DOF_E_2 – Mean and Mode

Similar to DOF_E_1, but the statistic computed and used for updates will be the mean on the train dataset.

The update for categorical features is equal o DOF_E_1, meaning that the mode is used.

5.2.6 DOF_F - Feature engineering

Feature engineering is the creation of features in the dataset, using the information that already exists. This can be a mathematical computation such as dividing a transaction amount by the number of transactions performed on the same day, some transformation on existing data, such as creating a feature that is 1 if the amount is higher than 1.000, and 0 otherwise.

As there is the need to perform a computation that is valid for all datasets, the option was to implement methods that already generate features based on datasets, as detailed below.

DOF_F_1 – No Feature Engineering

If this is the option chosen, then no new feature will be engineered.

DOF_F_2 – Add PCA

Principal Component Analysis, or PCA, is a technique that creates features to capture the variance in the dataset. In this option, four new features will be created with PCA, and will be added to the train, validation, and test datasets.

DOF_F_3 – Add LDA

One feature will be added, using Linear Discriminant Analysis. This new feature will be added to the existing datasets.

DOF_F_4 – Add PCA and then LDA

If this option is selected in DOF_F, then DOF_F_2 will be executed, and afterward, DOF_F_3 will also be executed. The new features will be added to the datasets.

5.2.7 DOF_G – Feature selection

It is undesirable to keep in the datasets features that do not contribute to the predictive capabilities of our model. Those features will not be adding value and will, at the very least, cause the processing to take longer.

They can also hinder the model's performance, because a high number of features might also lead to the "curse of dimensionality". This occurs when, by having a large number of features, the multidimensional space in which they are represented becomes very sparse, and therefore it is difficult to find observations that have small distances between them.

Some features may also cause overfitting, meaning that the model will adapt excessively to the train data, and instead of recognizing the patterns that determine the target classes, will learn to recognize the train instances individually. Then, when applied to new and unseen data, the model will not generalize properly (it will fail to make predictions).

DOF_G_1 – No Feature Selection

This corresponds to keeping the *status quo*. If this is the option for DOF_G, then all features are kept in the train, validation and test datasets.

DOF_G_2 – SelectKBest

SelectKBest algorithm is used to determine the best features. The top 80% features are kept; the worst 20% are dropped.

DOF_G_3 – Recursive Feature Elimination (RFE)

RFE is used to determine the best features. The top 80% features are kept; the worst 20% are dropped.

DOF_G_4 - Custom

A customized method is used to determine what features to keep and what features to drop. This involves performing tests for categorical and numerical values, assigning weights to each test, and comparing the result, feature by feature, with a defined threshold. Features that do not meet that threshold are dropped from all datasets (train, validation, and test).

5.2.8 DOF_H - Data Imbalance

It is common in supervised learning projects for the dataset to be imbalanced. Fraud detection is one of those cases, with positive labels accounting frequently for less than 1% of the total transactions. It is therefore relevant to implement strategies to compensate for this disparity.

Data imbalance strategies are performed exclusively on the train dataset. The validation and test dataset remain unaltered because they represent the reality in which the models will be operating. By changing the validation and test datasets, the measures would not be assessing the reality of the business problem.

DOF_H_1 - No Data Imbalance correction

Perform the analysis without balancing the number of instances for each class on the train dataset.

DOF_H_2 - RandomOverSampler

The minority class is enlarged by repeatedly selecting observations in this class, until the number of observations in both classes is approximately equal.

DOF_H_3 – RandomUnderSampler

The majority class is reduced by selecting just part of the observations in this class, until the number of observations in both classes is approximately equal.

5.2.9 DOF_I - Model

After all preprocessing is conducted on the data, the supervised model is the element that will analyze train data and search for the patterns and rules that will allow it to make predictions.

It is at this moment that the overall performance can be measured. When performing operations such as removing outliers, scaling data or compensating for data imbalance, it is not possible to immediately measure how much the predictive capability is improved or degraded. What needs to be done is to perform all the operations and then assess the model's results under that set of conditions.

It is therefore at this stage that results are obtained, recognizing that they depend on all previously taken steps, and also on the model that is being used.

Different models exist, with different strengths and weaknesses. This study looks at some of the models most used for fraud detection.

Even though it is important to optimize predictions by performing hyperparameter tuning in the typical supervised learning project, given the different datasets and the changes in the processing at each DOF, hyperparameter tuning would make it harder to evaluate the merits of each option. Therefore, the base parameters will be the ones employed for all models.

DOF_I_1 – LightGBM, default parameters

Light Gradient boosting machine is an algorithm that is popular in fraud detection (A. Aslam & Hussain, 2024). It was created by Microsoft to improve on memory usage and performance. By making some assumptions and simplifications that have small or no impact on the results, it can significantly reduce computational costs when compared to XGBoost.

Default parameters for LightGBM are presented in Appendix A.

DOF_I_2 – XGBoost, default parameters

Extreme Gradient Boosting is a top-performing algorithm that takes advantage of the benefits of Decision Trees and addresses some of the issues, such as overfitting, by introducing a regularization term that penalizes complex tree models. This model creates several trees, with each tree focusing on the observations that were not correctly predicted by the previous tree.

This makes it more likely that a misclassification will be correctly predicted in the following iteration. It is, however, relatively slow in execution.

Default parameters for XGBoost are presented in Appendix A.

DOF_I_3 – Decision Tree, default parameters

Decision Trees tend to overfit if left to train unrestricted. Several techniques exist (e.g., pruning) that can reduce this problem. One of their key advantages is that they provide high interpretability.

Default parameters for Decision Tree are presented in Appendix A.

DOF_I_4 – Random Forest, default parameters

This ensemble model creates several Decision Trees and trains each with just a part of the available features, and then combines the results into a final prediction. It is usually a very well-performing model and retains some explainability.

Default parameters for Random Forest are presented in Appendix A.

5.3 PIPELINE REPRESENTATION

For systematization, the options can be represented in a table. Each pipeline is created by randomly choosing one of the options available in each line.

Table 5.1 - Pipeline representation, with all DOF

			Option				
			1	2	3	4	5
DOF	A	Dataset	Cred.card	Bank Acc.	BankSim	Forage	PaySim
	B	Errors	Upd w/Null	Drop row			
	C	Outlier	Ignore	Upd IQRx3	Upd IQRx1,5	Del IQRx1,5	
	D	Scaling	Ignore	MinMax	Standard	Robust	
	E	Nulls	Median	Mean			
	F	Feat eng	Ignore	PCA	LDA	PCA+LDA	
	G	Feat sel	Ignore	SelectKBest	RFE	Customized	
	H	Resampl	Ignore	Rand over	Rand under		
	I	Model	LightGBM	XGBoost	Decis.Tree	Rand.Forest	

5.4 METRICS

Several metrics are calculated. Even though there is a difference in cost between false positives and false negatives, there is still the need to strike a balance between these two undesirable results. Therefore, the option chosen was to measure F1_Score.

When calculating the F1 score, several approaches can be used. For binary classification problems, the main options are “binary”, in which the F1 score is calculated for just one class (typically, the positive class); “macro”, in which the F1 score is determined for each of the classes and the result will be the average of those values; “weighted”, which is similar to “macro”, but a weighted average is calculated. The weights are based on the class’s support. This is a common approach for imbalanced datasets, and it was the one chosen for the experimental study.

5.5 HOLDOUT INSTEAD OF K-FOLD CROSS-VALIDATION

It is customary to use cross-validation, and specifically k-fold cross-validation (Vanneschi & Silva, 2023, p. 129), to produce more robust results. This approach serves well the purpose of ensuring that results obtained are not biased by a data split but has the drawback that training and validation need to be performed several (k) times, and therefore the time and resources consumed to obtain each result will be significantly higher. It is known that cross-validation is especially relevant when the dataset is small and so, conversely, given that the smallest dataset employed for testing has 200.000 observations, the likelihood of a certain random split significantly affecting results is negligible - but never zero.

An alternative considered was using repeated random subsampling, or Montecarlo cross-validation (Dubitzky et al., 2007, p. 178). This method creates repeated random splits of the data, but rather than repeating the process for a predetermined number of iterations, after a minimum number of iterations, the variance of the results obtained is observed, and if it is already very low, no further tests are performed.

It was considered, nevertheless, that better estimates for specific configurations are less important than allowing as many different configurations as possible to be assessed. Instead of aiming to test exactly and exhaustively a smaller number of configurations, the option is to perform tests on as many configurations as possible. It is significantly better to test 3.000 different configurations just once than to test 30 times, but only 100 different configurations.

The robustness of the results will be obtained by experimenting each option in as many scenarios as possible. For this reason, the option was to use the holdout method and to test as many configurations as possible.

5.6 ANALYZING RESULTS

Results obtained through experimentation, by running more than 2.000 distinct combinations, were then analyzed using a Multilinear regression. The results are presented in the next chapter.

6 RESULTS AND DISCUSSION

6.1 TEST COVERAGE

While testing different pipelines, it was controlled that each individual option along the pipeline would be used a significant number of times so that the results would be statistically significant.

Table 6.1 - Test coverage for each option

		Option					
		1	2	3	4	5	Total
DOF	A	444	404	384	408	412	2052
	B	992	1060				2052
	C	523	508	509	512		2052
	D	501	525	503	523		2052
	E	1020	1032				2052
	F	514	517	514	507		2052
	G	539	513	498	502		2052
	H	668	713	671			2052
	I	526	542	504	480		2052

Note that while the total number for each DOF is equal, as it corresponds to the number of runs, these are not uniformly distributed among the options of each DOF. This is due to the fact that, at each run and for each DOF, the option incorporated in the pipeline was randomly chosen.

6.2 A REGRESSION TO ANALYZE RESULTS

To establish the impact of each decision on the dependent variable, a Simple Linear Regression model presents several advantages, such as the capacity to determine the relation between those variables, by estimating the impact on the dependent variable (the variable whose changes are being explained) of changes made in the value of the explanatory variable (the variable that is controlled in the experimental study). This impact is measured by the simple linear regression model, which will output both the weight of the impact and a p-value that informs the degree of confidence on that weight estimation.

A Simple Linear Regression model cannot be used for the experimental study, as it is used to determine the way in which a change in one explanatory variable affects the dependent variable. As there are several variables to study, this model is not adequate.

6.3 MULTIPLE LINEAR REGRESSION MODEL, ASSUMPTIONS

The Multiple Linear Regression (MLR) model removes the assumption of independence of the explanatory variables. It is, therefore, a better option for the experimental research being conducted.

In order to use the Multiple Linear Regression model, Gauss-Markov assumptions have to be verified (Wooldridge, 2019, p. 79):

MLR1. Linear in parameters

This assumption is in fact the definition of the model; it is required to be linear in parameters, and in fact it is so by construction:

$$\mathrm{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \mathrm{u}$$

Where $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ are the unknown parameters that the model will determine.

It is however still possible for the dependent variables to be constructed based on arbitrary functions such as logarithms and squares. These functions may not be linear but nevertheless, the restriction applies to the Beta parameters, and those are ensured to be linear in the way that the model is defined.

MLR2. Random sampling

At each run, each DOF is randomly chosen. This creates a batch of tests and results that are a randomly chosen subset of the entire population of all combinations and all possible runs with all those combinations.

So, indeed, the MLR is being executed on a random sample extracted from the population.

MLR3. No perfect collinearity

Perfect collinearity could arise from one independent variable being constant. This is something that is controlled and it is guaranteed not to happen.

It could also arise from an exact linear relationship between two or more variables. So, if one of the variables can be exactly calculated based on the remaining variables, this assumption would not be verified.

Applying this to the experimental study, this could happen in two ways; either if the choice at one DOF was dependent on the choice at another DOF, which never happens by implementation (all DOF options are randomized at each run), or else by unfolding a categorical variable into several dummy variables and not withdrawing one of those variables. This is something that is controlled when generating the MLR model, and at each DOF the first possible value is always being omitted, as part of the one-hot encoding explained below.

MLR4. Zero conditional mean

This assumption could occur in several ways, such as a model misspecification. An example would be the omission of a relevant variable, including a variable that is correlated to variables already in the model.

This is more likely to occur when using a simple linear regression model, and since the experiment is controlled, with other factors being fixed, it is also more probable that no correlation exists between the explanatory variables and the errors.

MLR5. Homoskedasticity

This requires that the variance of the error is the same, regardless of the value of the explanatory variables. Given that we have a large sample, we are under the conditions of the central limit theorem and are therefore also under this assumption.

MLR6. Normality

This requirement is relevant as it enables the use of tests on statistical significance.

6.4 MULTIPLE LINER REGRESSION, GAUSS-MARKOV THEOREM

By verifying these assumptions, the model is under the conditions of the Gauss-Markov theorem, implying that the estimations for $\beta_{0,}\ \beta_1,\ \beta_2, \ldots\ , \beta_k$ are not only unbiased but are in fact BLUE (the Best Linear Unbiased Estimator).

6.5 MULTIPLE LINEAR REGRESSION – INPUT

The input for the multilinear regression model comes from creating different supervised models, testing them against a dataset, and measuring results. Each execution's information is stored in MLFlow and constitutes one observation that will be inputted into the MLR model.

Table 6.2 - Sample of information recorded for each run (additional fields omitted)

Duration	Metric (val) F1 weight.	DOF								
		A	B	C	D	E	F	G	H	I
3.3s	.99906459	4	2	3	1	1	4	2	1	3
40.3s	.96579900	2	2	4	4	1	4	4	1	3
3.1s	.99710019	4	1	4	4	2	3	4	2	1
1.5m	.99955730	5	1	2	3	1	4	1	2	3
4.7s	.96116298	3	2	4	2	1	1	3	3	3
27.8s	.98721221	3	2	4	1	2	1	1	1	4
33.9s	.99965408	5	1	3	3	2	2	1	1	2
6.0s	.99343181	3	2	3	3	1	4	1	1	2

By independently randomizing the choices at each DOF, then executing the proceeding a large number of times, options at each DOF end up being connected to all options on the next DOF.

For the purposes of this analysis, we have one dependent variable, val_f1_weighted, and 9 explanatory variables (DOF_A through DOF_I).

Each variable represents a distinct, discrete choice on a category, rather than the choice of a value over a continuous numerical interval. For instance, DOF_A represents the choice of which available dataset to use, which is a discrete choice. A numerical choice could be, for instance, the percentage of the dataset to use.

Given that in fact all the explanatory variables are categorical, in order to use them in the multiple linear regression, they need to be one-hot encoded. Again, taking the case of DOF_A, which varies from 1 to 5 in a total of five possible values, four variables will be created – DOF_A_2, DOF_A_3, DOF_A_4 and DOF_A_5. Each variable will be set to 1 when its implied value is present in the categorical value and will be set to 0 otherwise. The absent variable, in this case DOF_A_1, is represented by all other variables being equal to 0.

The following table illustrates this example:

Table 6.3 - One-hot encoding of a categorical feature

Original variable	One-hot encoding of the variable			
DOF_A	DOF_A_2 (is DOF_A=2?)	DOF_A_3 (is DOF_A=3?)	DOF_A_4 (is DOF_A=4?)	DOF_A_5 (is DOF_A=5?)
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1

It is relevant to note that the linear regression will take as a baseline all the options that are omitted (so in the example, it would consider DOF_A=1), and for each variable that is visible (DOF_A_2, DOF_A_3, DOF_A_4 and DOF_A_5) will measure the impact of switching from the hidden value (the baseline) to that option.

6.6 MULTIPLE LINEAR REGRESSION – RESULTS

The multilinear regression provides results about the model and additional results that are broken down by the model's variables, as shown in the following table.

6.6.1 Information about the model

The multiple linear regression ran with a total of 2,052 observations, and with a total of 23 variables. These 23 variables result from the one-hot encoding of the categorical variables DOF_A through DOF_I.

Table 6.4 - Multilinear regression, model results

No.Observations:	2 052
DfResiduals:	2 028
DfModel:	23
R-squared:	0,245
Adj.R-squared:	0,236
F-statistic:	28,61
Prob(F-statistic):	2,23E-106
Durbin-Watson:	2,025
Jarque-Bera(JB):	428 067
Prob(JB):	0,000
Skew:	-6,939

The first finding is that the value for R^2, 0.245, is relatively low. This means that, of all variation in the dependent variable, the variables present in the model explain a little less than 25%.

The F-statistic of 28.61, associated with a p-value for the f-statistic that is extremely low, signifies that the model is better at predicting results than if the explanatory variables were to be removed. In practical terms, the model is better at predicting than using the average for the dependent variable.

The Durbin-Watson's test results allow for the conclusion that there is no autocorrelation in the dataset.

The Jarque-Bera test suggests that data is not normally distributed; the skewness different from zero implies that the data is not symmetrically distributed; given that the value is negative, it is skewed to the left.

This is not totally unexpected and merits further analysis. The fact that predicted results are very close to 1, and that the F1 score has a maximum of 1, originates that necessarily the distribution will be truncated at that point, which is more of a problem because it is very close to the mean value of 0.9864. One option to consider would be to use a different measure, even if it is fabricated, in order to avoid these impacts.

6.6.2 MLR – Results for the explanatory variables

Each of these variables corresponds to the options for each degree of freedom that do not coincide with the baseline.

The details for each independent variable of the model are presented in the table below:

Table 6.5 - Multilinear regression - result for variables

Variable	coef	std err	t	P>|t|	[0,025	0,975]
baseline	,9864	,007	134,454	,000	,972	1,001
DOF_A_2	-,0299	,005	-6,421	,000	-,039	-,021
DOF_A_3	,0169	,005	3,587	,000	,008	,026
DOF_A_4	,0011	,005	,232	,817	-,008	,010
DOF_A_5	,0099	,005	2,143	,032	,001	,019
DOF_B_2	,0003	,003	,096	,923	-,006	,006
DOF_C_2	,0023	,004	,550	,582	-,006	,011
DOF_C_3	,0018	,004	,439	,661	-,006	,010
DOF_C_4	-,0194	,004	-4,626	,000	-,028	-,011
DOF_D_2	,0039	,004	,937	,349	-,004	,012
DOF_D_3	,0024	,004	,573	,566	-,006	,011
DOF_D_4	,0011	,004	,263	,793	-,007	,009
DOF_E_2	,0100	,003	3,359	,001	,004	,016
DOF_F_2	,0181	,004	4,299	,000	,010	,026
DOF_F_3	,0136	,004	3,222	,001	,005	,022
DOF_F_4	,0239	,004	5,658	,000	,016	,032
DOF_G_2	-,0010	,004	-,233	,816	-,009	,007
DOF_G_3	,0001	,004	,022	,982	-,008	,008
DOF_G_4	-,0325	,004	-7,751	,000	-,041	-,024
DOF_H_2	-,0134	,004	-3,697	,000	-,021	-,006
DOF_H_3	-,0659	,004	-17,854	,000	-,073	-,059
DOF_I_2	-,0084	,004	-2,043	,041	-,017	,000
DOF_I_3	-,0076	,004	-1,811	,070	-,016	,001
DOF_I_4	,0034	,004	,793	,428	-,005	,012

Baseline results

The first line of the table, “baseline”, measures the performance of the model using all options that were one-hot encoded and are therefore absent in the explanatory variables on the lines below.

This configuration has a validation f1_score_weighted of 0.9864, which is extremely high, given that the F1 score varies between 0 and 1. The p-value associated with this result is

extremely low, meaning that there is very strong statistical evidence that it contributes to the model's capacity to predict results.

The fact that this result, 0.9864, is very close to the limit of 1, may account for the skewness to the left, as the distribution is truncated to the right, and very close to the value observed for the baseline. In fact, the critical area to the right starts at 1.001, which is already above the limit, meaning that it is certain that there will be no observations in that critical area.

The lines that follow show the predicted impact of changing from the baseline value to the corresponding value.

DOF_A – Dataset

Line DOF_A_2 on the table shows the impact of using the dataset 2 instead of the dataset that is part of the baseline (dataset 1, corresponding to DOF_A = 1). The fact that the impact is negative, specifically -0.0299, means that if dataset 2 is used instead of dataset 1, it is expected that the dependent variable (validation f1_score_weighted) will decrease by 0.0299. The p-value is represented as 0.000, which is a rounding of the actual value: 1.676231e-10. The interpretation is that it is extremely unlikely that moving from DOF_A=1 to DOF_A=2 would fail to produce changes.

Contrary to DOF_A_2, the remaining options for DOF_A (3, 4 and 5) all present positive values. This means that it is expected for the predicted variable to increase if the baseline dataset is replaced with any of these. However, this is in fact not the case for DOF_A=4, as the p-value is extremely high. This means that it would not be unlikely for DOF_A_1 and DOF_A_4 to have the same F1 score, and for a sample to produce data such as what was observed.

DOF_B – Error correction

DOF_B_2 presents an extremely high p-value, meaning that the prediction could easily be wrong. Additionally, the coefficient is extremely low, meaning that even if it was significant, it would not have a strong impact.

In practical terms, under the conditions of the experimental study, the option chosen for handling errors does not impact the model's performance.

This could be explained by the fact that the datasets presented are mostly free from errors, and also by the fact that the verification of errors was constructed manually, not inferred from the data. It is possible that either a more aggressive error detection approach or a lower-quality dataset would result in this process being more critical than it proved to be under the given circumstances.

DOF_C – Outlier handling

DOF_C = 1, the option that is embedded in the baseline, consists of ignoring outliers.

DOF_C_2 and DOF_C_3 involve replacing outliers with values that are less distant from the first and third quartile of the distribution. Both DOF_C_2 and DOF_C_3 present low coefficients and very high p-values. This means that both these strategies for handling outliers failed to prove that they are better than ignoring outliers.

DOF_C_4 consists in deleting outliers from the training dataset. Given the very low p-value of 3.963794e-06 (rounded to 0.000 on the summary table), the coefficient obtained is significant. However the coefficient is negative, so the conclusion is that we are confident that deleting outliers reduces the f1-score for the model.

DOF_D – Scaling

All presented options, DOF_D_2 (MinMaxScaler), DOF_D_3 (StandardScaler) and DOF_D_4 (RobustScaler) present high p-values, and therefore we have no confidence in the coefficient obtained. This means that these strategies did not prove to be better than DOF_D=1, the option presented on the baseline, that consists in not scaling data.

Several factors are likely to contribute cumulatively to this result, which at first seems to be disconcerting. One aspect is that datasets with several categorical features, or with features varying in a restricted range, will be less impacted by poor or no scaling. Furthermore, it is noteworthy that the classification models employed are all tree-based models, and these are the models that are less affected by scaling.

Some further research might be conducted in the future to verify this interpretation.

DOF_E – Null value handling

Null values in categorical features are in all cases being replaced with the mode for that category, determined on the train dataset.

In the baseline, null values in numerical features are replaced with the median for that feature, calculated on the train dataset. This corresponds to DOF_E=1.

DOF_E_2 uses, for numerical features, the mean instead of the median. This option improves the F1 score, the predicted variable, by 0.01. The very low p-value of 0.001 shows that this improvement is trustworthy.

DOF_F – Feature engineering

The feature engineering on the baseline corresponds to maintaining just the variables in the dataset.

All alternatives to this strategy show a positive coefficient (DOF_F_2=0.0181, DOF_F_3=0.0136 and DOF_F_4=0.0239) and have an extremely low p-value, so the results are significant and these are better strategies than the baseline option.

Given that strategy 2 consists in creating 4 features with PCA, strategy 3 in creating one feature with LDA, and strategy 4 consists in using both previous strategies sequentially, it seems logical that DOF_F_4 would provide the best improvement, as it combines two strategies that each yield a positive result. The data supports this.

DOF_G – Feature selection

DOF_G_2 and DOF_G_3 present high p-values and therefore don't seem to have a significant impact on the model.

Contrary to that, DOF_G_4 has an extremely low p-value, associated with a very negative coefficient. This means that using it will cause worse results than not using it.

DOF_G_4 is the feature selection that is customized to use several criteria, combine them, and arrive at a final decision on what features to keep and what features do drop. As a customized function, it could be optimized to obtain better results. What is established from the results is that, as it is, its use is just hindering results.

DOF_H – Data imbalance

The baseline's strategy for data imbalance is to simply ignore it. The alternative strategies are DOF_H_2 (random over sampling) and DOF_H_3 (random under sampling).

Intuitively, ignoring the data imbalance is an option that does not make sense, as the datasets analyzed are extremely imbalanced.

It is therefore unexpected that both DOF_H_2 and DOF_H_3 have a worse performance than the baseline (respectively, -0.0134 and -0.0659). The p-values associated are extremely low, meaning that we have strong confidence that indeed these strategies cause worse results.

This is likely to be caused by the choice of the dependent variable; f1 score weighted assigns weights proportionally to the observed ratio of observations. By correcting the imbalance, the model will have more observations from the minority class to analyze, and it is possible that it is less competent at detecting that class.

Another hypothesis is that the imbalance correction was excessive, and the results could be different with a different ratio, such as 90/10 or 85/15 between the negative and the positive classes. This could be implemented as additional options in DOF_H.

Finally, more sophisticated approaches to correct data imbalance could be employed, such as SMOTE or ADASYN.

DOF_I – Models

The baseline model is LightGBM.

Of the alternatives considered, DOF_I_2 is XGBoost; its use is expected to lower the f1 score by 0.0084, and the p-value of 0.041 means that we have confidence in this result.

DOF_I_3 is a Decision Tree, whose performance is also estimated to be worse, however the p-value is above the usual threshold of 0.05, meaning that there is not much confidence in this result. DOF_I_4 is a Random Forest. The p-value for this estimate is extremely high, meaning that it must be disregarded.

Considering the good results obtained by LightGBM, which is the model used in the baseline, and the fact that it trains and predicts quicker than the alternatives analyzed, it is understandable why it is commonly used in fraud detection systems.

6.7 RESULTS - BUSINESS IMPACT

Using the results obtained, it is possible to reassess the pipeline and signal if an option contributed positively or not to the results. Ultimately, this can also be evaluated at the DOF level, i.e., which steps have a positive impact, and which do not.

This is the representation of such impacts on our initial pipeline, as described before. The purpose is to identify options that have a positive contribution; options that have a negative impact or whose impact is not statistically significant, even if presented as positive, can be removed to simplify the pipeline.

Table 6.6 - Pipeline: Assessing the impact of each option

			Option				
			1	2	3	4	5
DOF	A	Dataset	Cred.card	Bank Acc.	BankSim	Forage	PaySim
	B	Errors	Upd w/Null	Drop row			
	C	Outlier	Ignore	Upd IQRx3	Upd IQRx1,5	Del IQRx1,5	
	D	Scaling	Ignore	MinMax	Standard	Robust	
	E	Nulls	Median	Mean			
	F	Feat eng	Ignore	PCA	LDA	PCA+LDA	
	G	Feat sel	Ignore	SelectKBest	RFE	Customized	
	H	Resampl	Ignore	Rand over	Rand under		
	I	Model	LightGBM	XGBoost	Decis.tree	Rand.forest	

Given that DOF_A is the choice of a dataset, in practice the data scientist will not be choosing to use one dataset over another dataset; therefore, no selection is being performed at that level.

In several DOF, the alternatives to the baseline are all worse (or at least, not significantly better) than the baseline option. For those, it makes sense to remove these options. It so happens that, in some cases, the baseline option is a certain form of processing (Nulls: Median;

Model: LightGBM) for others, it is just “Ignore” (Outlier, Scaling). When only the baseline option remains, and that option is to ignore the step, we can remove that DOF altogether.

We get therefore a much simplified version of the baseline. By removing Outlier handling, scaling, feature selection and resampling, and also removing some of the options in the remaining DOF, we get this streamlined version of the original pipeline, which is expect to perform as well as the initial:

Table 6.7 - Pipeline reconstructed from the experimentation results

			Option				
			1	2	3	4	5
DOF	A	Dataset	Cred.card	Bank Acc.	BankSim	Forage	PaySim
	B	Errors	Upd w/Null				
	E	Nulls		Mean			
	F	Feat eng		PCA	LDA	PCA+LDA	
	I	Model	LightGBM				

To analyze the impact of this significant simplification, we can reassess the number of distinct pipelines that we can build. From the initial account of 61.440 unique pipelines, now the experimentation can be restricted to

5 * 1 * 1 * 3 * 1 = 15

Brute force experimentation on these 15 pipelines, resorting to repeated k-fold cross validation with 30 runs, would require a mere 450 runs.

As what is being proposed is a methodology to compare results, not a set of definitive conclusions, it is extremely important to emphasize that more complete runs of more complex pipelines should still be conducted, to account for changes within the pipeline (for instance, the introduction of models that are more sensitive to outliers may require the reintroduction of that DOF). Even if there are no internal changes, exogenous factors such as data drift and concept drift, that are well known to occur in the domain of fraud, require the reassessment of conclusions obtained at a certain moment and context.

6.8 Results - Further exploration

All the analyses performed so far combine all options for all DOFs.

In practice, the data scientist might be interested in further understanding the impacts of the options in a specific context. As an example, it might be relevant to explore the impacts of all DOF when the model is a Decision Tree, or even to explore the impacts for each supervised model, in order to determine if different models are impacted in different ways by some of the options. This can easily be achieved by filtering data just for one model, then run the multilinear regression for that subset of data.

When looking at the DOFs defined in the experimentation, which represent the options available for the data scientist, arguably the choice of the dataset is not a real-world option; the pipelines are run with the data available, and it is not really possible to change the dataset altogether to get better results.

Taking this into consideration, the multilinear regression was run again, but filtering the runs and results for each dataset. This means that, in practice, DOF_A was no longer interpreted as a degree of freedom, and was therefore removed from the table. For simplicity, the table presents only the column of coefficients for each Dataset.

The first immediate observation is that the R^2 improves significantly for all datasets. It also becomes apparent that the options that are more impactful will vary. In the table below, green and red cells are those whose coefficients are statistically significant, either positive (green) or negative (red).

Table 6.8 - Results broken down by DOF_A

	A_1	A_2	A_3	A_4	A_5
R^2	0,218	0,709	0,816	0,550	0,425
baseline	1,0168	,9598	,9929	,9866	,9754
DOF_B_2	,0187	-,0015	,0002	,0026	-,0097
DOF_C_2	,0065	,0021	-,0017	,0015	-,0013
DOF_C_3	,0037	,0072	-,0025	,0028	-,0013
DOF_C_4	-,0769	,0311	-,0127	-,0149	-,0112
DOF_D_2	,0012	,0031	,0002	,0053	,0079
DOF_D_3	,0046	-,0051	-,0011	,0055	-,0001
DOF_D_4	,0026	,0005	,0007	,0084	-,0016
DOF_E_2	,0206	-,0005	,0006	,0124	,0123
DOF_F_2	-,0068	,0168	,0028	,0138	,0533
DOF_F_3	-,0270	,0092	,0020	,0244	,0523
DOF_F_4	,0023	,0128	,0016	,0277	,0561
DOF_G_2	,0085	,0012	,0003	-,0088	-,0028
DOF_G_3	,0058	,0035	-,0010	-,0079	,0020
DOF_G_4	-,0321	-,0212	-,0026	-,0348	-,0659
DOF_H_2	-,0029	-,0329	-,0047	-,0019	-,0301
DOF_H_3	-,0828	-,1026	-,0229	-,0860	-,0309
DOF_I_2	-,0349	,0089	,0021	-,0061	-,0010
DOF_I_3	-,0223	-,0029	,0036	-,0091	-,0083
DOF_I_4	-,0027	,0190	,0029	-,0009	,0032

This table shows that some options tend to have positive impact, negative impact or have an impact that is not statistically significant. It seems that, in fact, breaking down the results by dataset will allow for a better understanding of the problems.

Taking the Dataset corresponding to DOF_A_3, which has the highest R^2 value, we see that it has the worst results when using LightGBM (the baseline model). It is also the only dataset that is penalized by all approaches implemented for outlier handling. Even if in both cases the actual coefficient is small in absolute value, it nevertheless stands out as different from the remaining datasets.

We can reconstruct the table that shows the impact of each option, now just for this dataset:

Table 6.9 - Pipeline for DOF_A_3: Assessing the impact of each option

			Option				
			1	2	3	4	5
DOF	A	Dataset			BankSim		
	B	Errors	Upd w/Null	Drop row			
	C	Outlier	Ignore	Upd IQRx3	Upd IQRx1,5	Del IQRx1,5	
	D	Scaling	Ignore	MinMax	Standard	Robust	
	E	Nulls	Median	Mean			
	F	Feat eng	Ignore	PCA	LDA	PCA+LDA	
	G	Feat sel	Ignore	SelectKBest	RFE	Customized	
	H	Resampl	Ignore	Rand over	Rand under		
	I	Model	LightGBM	XGBoost	Decis.tree	Rand.forest	

This will lead to this streamlined version:

Table 6.10 - Pipeline for DOF_A_3, reconstructed from the experimentation results

			Option				
			1	2	3	4	5
DOF	A	Dataset			BankSim		
	B	Errors	Upd w/Null				
	E	Nulls	Median				
	F	Feat eng		PCA	LDA		
	I	Model		XGBoost	Decis.tree	Rand.forest	

In this case, there will be merely 6 different pipeline configurations:

1 * 1 * 1 * 2 * 3 = 6

7 CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

Machine Learning models are widely used for fraud detection in electronic transactions. These models require several options to be considered by the data scientist at different steps along the sequential process. These options constitute the Degrees of Freedom (DOF) that are available for the data scientist to tweak the whole system.

The aim of the thesis is to measure the individual impacts of the decisions made at each DOF on the system's overall performance.

To attain this objective, a dynamic application was developed, establishing nine sequential steps. At each of these, a random option is automatically chosen, from a subset of implemented alternatives. Depending on the step, there are between two and five available implementations. The application was built in such a way that all options are compatible. In fact, more implementations can be created, as long as the structure of both expected inputs and outputs is kept.

An extensive experimentation was conducted, with different pipelines, with each run using a randomly selected combination of options and generating a result that was then recorded. More than two thousand executions were conducted, registering the options chosen, which constitute the explanatory or independent variables, and the results, which constitute the target or dependent variable.

The resulting dataset was then analyzed by what could be interpreted as a meta-model, in the sense that it is operating on top of the results produced by the models employed. The Multilinear Regression (MLR) was the model chosen for this analysis.

An MLR creates a baseline configuration and then calculates the difference between each option and that baseline. This baseline is created considering, for categorical variables, the value that is omitted when using one-hot encoding. As for numerical variables, they are considered to be 0 in the baseline; however, our dataset does not have numerical explanatory variables.

By achieving an extremely low p-value of 2.23e-106 on the F-statistic, we reject the Null hypothesis (that the regression is not meaningful). This proves that the regression provides significantly better estimates than using the mean. However, the R^2 value of 0.245 tells us that the model only explains 24.5% of the variance of the independent variable. This however is not a reason to discard the model, because as will be shown several variables are statistically significant. We obtain a skewness of -6,939. This means the distribution of results is not symmetrical, as this value is not zero, and that it’s skewed to the left because the value is negative.

The MLR also determines the impact of changing the value of each variable. Given that we're using one-hot encoding for all variables, as all of them are categorical, changing the value corresponds to changing the option to be used in the pipeline. For each variable, the MLR calculates its contribution to the result, positive or negative, and determines the statistical significance of such calculation. A result is considered significant when the p-value for that result is 0.05 or smaller.

Close to half of the options presented have a difference that is not statistically significant from the option in the baseline.

Of the results that are statistically significant DOF_A, that relates to the datasets employed, shows that several alternatives are significantly better or worse than the baseline. Additionally, the coefficients are some of the highest obtained. This signals the relevance of the data that is analyzed; all other things considered, some datasets provide better results than others.

Outlier handling and scaling strategies did not affect the result, except for DOF_C_4, that is significant and has a negative coefficient. This tells us that both ignoring outliers, which is the baseline strategy for outliers, or updating the outliers, does not make a relevant difference. However, deleting observations that have outliers does affect adversely the capacity of the model to produce good predictions.

For the imputation of Null Values on numeric variables, replacing with the mean (DOF_E_2) works better than replacing with the Median, which was the baseline option.

All techniques used for feature engineering improved results: DOF_F_2 creates 4 features using Principal Component Analysis (PCA), and DOF_F_3 generates one feature using Linear Discriminant Analysis. DOF_F_4 is the combination of both, PCA(4)+LDA(1). It seems intuitive, and is demonstrated in the results, that by combining two alternatives that yield good results, will yield an even better result.

In feature selection, the only option that has a statistically significant impact is a customized function for feature selection that is much worse than the alternatives or, in fact, worse than using no feature selection, which is the baseline option. The custom function that produced these poor results has a specific set of techniques and tests, some fixed thresholds for rejection, and criteria for the final vote; it is possible that another combination of techniques and thresholds would yield good results; the conclusion that it did not help the model is limited to the specific combination used.

As for addressing dataset imbalance on the train dataset, both random undersampling and random oversampling are worse than using the data without compensation techniques to correct the data imbalance.

The baseline supervised learning model, LightGBM, compares favorably with the alternatives. Its results are not statistically different from a Random Forest, however, on average the results are obtained in a small fraction of the time taken by a Random Forest to train. XGBoost gives worse results than LightGBM. Decision Trees also obtain worse results, with a p-value of 0.070 that is not significant given the threshold defined of 0.05.

To sum up, we can say that some factors are key to obtaining optimal results, starting with collecting good data and then improving it with techniques such as feature engineering. Some other techniques show less significance but, while there might not be substantial space for improvement at those levels, it is nevertheless possible, due to poor implementation, to seriously hinder results. This means that they cannot be altogether ignored.

It is relevant to remember that, in the effort to reduce the expert interference in the experiment, most of what was used was deliberately not customized or fine-tuned; models were trained with their default sci-kit learn parameters, as was scaling, dataset resampling, and so on. By fine-tuning each of these elements, it would very likely be possible to obtain better results.

If at first sight the supervised models showed similar results, other relevant factors such as processing time show that not all models are created equal. Some configurations are bound to obtain very good results in a fraction of the time.

This thesis builds upon strong foundations that exist in the literature, namely for choosing the options made available at each DOF. On top of this, it proposes an innovative approach that aims to facilitate understanding the impact of certain options and configurations on the overall model used for fraud detection, in a manner that is explainable and actionable, from a business perspective.

Some very straightforward observations may be conducted and provide opportunities for improvement on models running in production. One example would be determining that a certain option is significantly better than all the others, so possibly we can resort to that option, and there is no longer the need to keep the alternatives. Another example would be that we determine that for a certain DOF, all options provide poor results. Eventually, this DOF can be altogether removed from the pipeline.

A more sophisticated analysis could be that a certain option significantly degrades train time but does not significantly improve the predictive capability of the model. Possibly it can also be removed from the pipeline, or executed in a different order.

If the result for the F1 score is not satisfactory, it is easy to analyze separately the two components of the F1 score, precision and recall, provided that they were also logged. The data scientist can easily determine which techniques are penalizing through a reduced precision and which are penalizing through a reduced recall.

As shown, this approach has the capacity to quickly signal impacts and identify valid possibilities to address them, in a structured and consistent manner, rather than executing millions of combinations, working with reduced datasets or resorting to opinions on what changes will have more impact.

When transitioning from a more conceptual approach to a real-world situation, it is expected that the dataset will not actually constitute a DOF. When filtering per dataset, the results became more interesting, as shown by the significant increase in R^2 for all datasets, and by the fact that different options show different impacts.

7.2 FUTURE WORK

The approach explored in this document can be generalized to study pipelines for fraud detection, or in fact for other supervised learning problems.

It will be relevant to study the impact when using different metrics, such as the F1 macro or, possibly more interesting, using custom loss functions. One of such loss functions could include an estimate of the actual financial cost for false positives and false negatives, which depends significantly on the amount of the transaction. Another venue to explore is the processing cost and the responsiveness in predictions, which could be approximated through the train time and through the time taken to predict the label for a new observation. These measures will be important for the models to be usable in a production, real-time environment. These metrics can be easily collected during tests and afterward it is possible to analyze how a certain metric is composed, by simply setting the desired metric as the dependent variable in the MLR and checking the contribution of each option in each DOF.

The impact of the dataset imbalance and the measures to compensate for it, which ended up degrading the results, also need to be further studied, possibly in connection with the analysis of different metrics or the use of different techniques. Moreover, it would be possible to implement an additional DOF that would establish the level of imbalance correction (with options such as “unchanged”, “high imbalance”, “medium imbalance” and “no imbalance”).

Only tree-based models were used, and this has significant consequences, specifically in the scaling and outlier handling. As Decision Trees (and derived models such as Random Forests, XGBoost, and LightGBM) use Information Gain and Gini index to fit the train data, rather than measures based on some form of distance, these models become agnostic to the scaling of numeric data. So, another venue to explore is the use of some algorithms that are sensitive to the scaling of data, such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN).

One area to develop is feature engineering, that was shown to have a positive and significant impact. By limiting feature engineering to using Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), it was ensured that the techniques were not dependent on *a priori* knowledge about the dataset. However, there are more sophisticated tools that can be

employed for feature engineering, such as polynomial combinations of features or even clustering.

Hyperparameter tuning (HPT) was not included, as it was perceived that it could bias results towards a model that was better optimized. Nevertheless, this is a fundamental part of a Machine Learning project and, therefore, should be carefully researched. This can be implemented as an additional DOF, with options such as Optuna, Bayes search or an optimization using genetic algorithms. It should be interesting to evaluate the contribution of the HPT techniques on the metric, while controlling for the impact on training time.

There was a deliberate effort for the analysis not to be dependent on the expertise or prior knowledge about the datasets and models, as it could happen that a certain option would get better results due to superior optimization. This opens opportunities for future investigation, possibly taking advantage of existing research and knowledge on AutoML.

The ideas previously mentioned can be pursued in a manner that is consistent with what was used in this thesis, as they mostly require either creating new configurations for the pipeline or using the MLR for different analyses of the data. It is also possible to extend this approach and address broader questions, as follows.

The meta-model used, an MLR, can be substituted by a non-linear alternative that could provide further insights into the results, not parting from the need to address explainability and measure the statistical significance of the results.

Also relevant is the study on how to move from these results, that are based on decisions at each step, to modeling recommendations for a complete pipeline, for use in a production environment. In such environment, either a single or a very small number of pipelines will be responsible for analyzing data and making predictions.

Moreover, while the implementation mostly assumed defaults and performed no tweaking, for some specific cases, there was an unsurpassable need to define thresholds or parameters. In prolonging the analysis performed, it would be desirable to experiment with different values for these parameters, obtaining new results and hopefully new insights. The traditional approach would be to change classification thresholds or to create a custom loss function, but in coherence with the research presented, this could also be implemented as a new Degree of freedom with several options, with the multilinear regression then assessing which of the options performs best.

As several of these suggestions part away from one of the key decisions that was established in the thesis, that the analysis should not be biased by the skills of the data scientist or by the knowledge about a specific dataset, some interesting questions can be considered, such as:

Can a general-purpose AutoML model be surpassed by an AutoML model specialized in a certain domain, such as fraud detection?

How does a good data scientist compare to an AutoML-created model?

Who gets better results, and by what margin?

It is my personal opinion that AutoML will be able to simplify and automate the early stages of ML, and that field experts can still contribute significantly to the interpretation of results in the context of the problem, to the optimization vis-à-vis the business needs, and possibly to ensure that AI remains ethical.

This combination of the best capabilities of each part may be the path to excellence.

INDEX

BIBLIOGRAPHICAL REFERENCES

Abedin, M. Z., Chi, G., Uddin, M. M., Satu, Md. S., Khan, Md. I., & Hajek, P. (2021). Tax Default Prediction Using Feature Transformation-Based Machine Learning. *IEEE Access*, *9*, 19864–19881. https://doi.org/10.1109/ACCESS.2020.3048018

Abedin, M. Z., Hajek, P., Sharif, T., Satu, Md. S., & Khan, Md. I. (2023). Modelling bank customer behaviour using feature engineering and classification techniques. *Research in International Business and Finance*, *65*, 101913. https://doi.org/10.1016/j.ribaf.2023.101913

Acuna, E., & Rodriguez, C. (2013). A Meta analysis study of outlier detection methods in classification. *International Journal of Engineering Research & Technology*.

Adepoju, O., Wosowei, J., lawte, S., & Jaiman, H. (2019). *Comparative Evaluation of Credit Card Fraud Detection Using Machine Learning Techniques* (p. 6). https://doi.org/10.1109/GCAT47503.2019.8978372

Ai, X., Wu, J., Sheng, V., Zhao, P., & Cui, Z. (2015). Immune Centroids Oversampling Method for Binary Classification. *Computational Intelligence and Neuroscience*, *2015*, 109806. https://doi.org/10.1155/2015/109806

Akers, M. D., & Gissel, J. L. (2006). *What Is Fraud and Who Is Responsible?*

Alexopoulos, P., Kafentzis, K., Benetou, X., Tagaris, A., & Georgolios, P. (2007). *Towards a Generic Fraud Ontology in e-Government.* 269–276.

Anand, A., Pugalenthi, G., Fogel, G. B., & Suganthan, P. N. (2010). An approach for classification of highly imbalanced data using weighting and undersampling. *Amino Acids*, *39*(5), 1385–1391. https://doi.org/10.1007/s00726-010-0595-2

Anggoro, D. A., & Mukti, S. S. (2021). Performance Comparison of Grid Search and Random Search Methods for Hyperparameter Tuning in Extreme Gradient Boosting Algorithm to Predict Chronic Kidney Failure. *International Journal of Intelligent Engineering and Systems*, *14*(6), 198–207. https://doi.org/10.22266/ijies2021.1231.19

Aslam, A., & Hussain, A. (2024). A Performance Analysis of Machine Learning Techniques for Credit Card Fraud Detection. *Journal on Artificial Intelligence*, *6*, 1–21. https://doi.org/10.32604/jai.2024.047226

Aslam, F., Hunjra, A. I., Ftiti, Z., Louhichi, W., & Shams, T. (2022). Insurance fraud detection: Evidence from artificial intelligence and machine learning. *Research in International Business and Finance*, *62*, 101744. https://doi.org/10.1016/j.ribaf.2022.101744

Ayele, W. Y. (2020). Adapting CRISP-DM for Idea Mining A Data Mining Process for Generating Ideas Using a Textual Dataset. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, *11*(6), 20–32.

Baesens, B. (2022). Fraud analytics: A research agenda. *Journal of Chinese Economic and Business Studies*, *21*. https://doi.org/10.1080/14765284.2022.2162246

Baesens, B., Höppner, S., Ortner, I., & Verdonck, T. (2021). robROSE: A robust approach for dealing with imbalanced data in fraud detection. *Statistical Methods & Applications*, *30*. https://doi.org/10.1007/s10260-021-00573-7

Baesens, B., Höppner, S., & Verdonck, T. (2021). Data engineering for fraud detection. *Decision Support Systems*, *150*, 113492. https://doi.org/10.1016/j.dss.2021.113492

Banco de Portugal (Director). (2020, May 13). *O fabrico da moeda metálica: Da antiguidade aos dias de hoje.* [Video recording]. https://www.youtube.com/watch?v=p4OSrjDmbFI

Barnett, V., & Lewis, T. (1998). *Outliers in statistical data* (3ed ed.). Wiley.

Barua, S., Islam, M., & Murase, K. (2011). *A Novel Synthetic Minority Oversampling Technique for Imbalanced Data Set Learning* (Vol. 7063, p. 744). https://doi.org/10.1007/978-3-642-24958-7_85

Bauder, R. A., Khoshgoftaar, T. M., & Hasanin, T. (2018). An Empirical Study on Class Rarity in Big Data. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 785–790. https://doi.org/10.1109/ICMLA.2018.00125

Bergstra, J., & Bengio, Y. (2012). *Random Search for Hyper-Parameter Optimization*.

Bockel-Rickermann, C., Verdonck, T., & Verbeke, W. (2023). Fraud analytics: A decade of research. *Expert Systems with Applications*, *232*, 120605. https://doi.org/10.1016/j.eswa.2023.120605

Boel, P. (2019). Payment systems – history and challenges. *PAYMENT SYSTEMS*. https://www.riksbank.se/globalassets/media/rapporter/pov/artiklar/engelska/2019/190613/er-2019_1-payment-systems--historical-evolution-and-literature-review.pdf

Bolton, R. J., & Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, 235–255.

Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, *17*(3), 235–255. https://doi.org/10.1214/ss/1042727940

Bramer, M. (2016). *Principles of Data Mining*. Springer London. https://doi.org/10.1007/978-1-4471-7307-6

Carcillo, F., Le Borgne, Y.-A., Caelen, O., & Bontempi, G. (2018). Streaming active learning strategies for real-life credit card fraud detection: Assessment and visualization. *International Journal of Data Science and Analytics*, *5*, 285–300.

Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, *557*, 317–331. https://doi.org/10.1016/j.ins.2019.05.042

Catley, C., Smith, K., Mcgregor, C., & Tracy, M. (2009). Extending CRISP-DM to incorporate temporal data mining of multidimensional medical data streams: A neonatal intensive care unit case study. In *Proceedings—IEEE Symposium on Computer-Based Medical Systems* (p. 5). https://doi.org/10.1109/CBMS.2009.5255394

Caton, J. L., & Harwick, C. (2022). Cryptocurrency, Decentralized Finance, and the Evolution of Money: A Transaction Costs Approach. *Journal of New Finance*, *2*(4). https://doi.org/10.46671/2521-2486.1027

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature Engineering Strategies for Credit Card Fraud Detection. *Expert Systems with Applications*, *51*. https://doi.org/10.1016/j.eswa.2015.12.030

Cotton, D. L., Johnigan, Sandra, & Givarz, L. (2023). *Fraud Risk Management Guide: Executive summary*.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). *Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information*. https://doi.org/10.1109/IJCNN.2015.7280527

de Castro, L. N., & Timmis, J. I. (2003). Artificial immune systems as a novel soft computing paradigm. *Soft Computing*, *7*(8), 526–544. https://doi.org/10.1007/s00500-002-0237-z

Destefanis, G., Barge, M. T., Brugiapaglia, A., & Tassone, S. (2000). The use of principal component analysis (PCA) to characterize beef. *Meat Science*, *56*(3), 255–259. https://doi.org/10.1016/S0309-1740(00)00050-4

Devi, D., Biswas, S., & Purkayastha, B. (2019). *A Cost-sensitive weighted Random Forest Technique for Credit Card Fraud Detection* (p. 6). https://doi.org/10.1109/ICCCNT45670.2019.8944885

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, *465*, 1–20. https://doi.org/10.1016/j.ins.2018.06.056

Dubitzky, W., Granzow, M., & Berrar, D. (2007). *Fundamentals of Data Mining in Genomics and Proteomics*. Springer. https://link.springer.com/book/10.1007/978-0-387-47509-7

Fan, X., Tang, K., & Weise, T. (2011). *Margin-Based Over-Sampling Method for Learning from Imbalanced Datasets* (Vol. 6635, p. 320). https://doi.org/10.1007/978-3-642-20847-8_26

Fanai, H., & Abbasimehr, H. (2023). A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection. *Expert Systems with Applications*, *217*, 119562. https://doi.org/10.1016/j.eswa.2023.119562

Fonseca, J., Douzas, G., & Bação, F. (2021). Improving Imbalanced Land Cover Classification with K-Means SMOTE: Detecting and Oversampling Distinctive Minority Spectral Signatures. *Information*, *12*, 266. https://doi.org/10.3390/info12070266

Forage. (2023). [Dataset]. https://www.theforage.com/simulations

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 1–37. https://doi.org/10.1145/2523813

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). *Why do tree-based models still outperform deep learning on tabular data?* (arXiv:2207.08815). arXiv. http://arxiv.org/abs/2207.08815

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, *3*(null), 1157–1182.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed). Elsevier.

Hand, D. J., Whitrow, C., Adams, N. M., Juszczak, P., & Weston, D. (2008). Performance criteria for plastic card fraud detection tools. *Journal of the Operational Research Society*, *59*(7), 956–962.

Hassan, N., Altiti, O., Abu Aqouleh, A., & Younes, M. (2020). *Credit Card Fraud Detection Based on Machine and Deep Learning* (p. 208). https://doi.org/10.1109/ICICS49469.2020.239524

Hawkins, D. M. (1980). *Identification of Outliers. Monographs on Applied Probability and Statistics.* (D. M. Hawkins, Ed.). Springer Netherlands. https://doi.org/10.1007/978-94-015-3994-4_1

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

Heath, J., & Mcgregor, C. (2010). *CRISP-DM0: A method to extend CRISP-DM to support null hypothesis driven confirmatory data mining.* (p. 101).

Hegde, D. S., Samanta, D., & Dutta, S. (2022). Classification Framework for Fraud Detection Using Hidden Markov Model. In J. M. R. S. Tavares, P. Dutta, S. Dutta, & D. Samanta (Eds.), *Cyber Intelligence and Information Retrieval* (pp. 29–36). Springer. https://doi.org/10.1007/978-981-16-4284-5_3

Heron, T. (2017). *Do you want payment in Iron or Salt? The economic basis of societal development in the European Iron Age*.

Hicks, J. R. (1962). Liquidity. *The Economic Journal*, *72*(288), 787–802. https://doi.org/10.2307/2228351

Hodge, V., & Austin, J. (2013). A Survey of Outlier Detection Methodologies (Reprint). In *Artificial Intelligence Review—AIR* (Vol. 22).

Horesh, N. (2012). From Chengdu to Stockholm: A Comparative Study of the Emergence of Paper Money in East and West. *Provincial China*, *4*(1), Article 1. https://epress.lib.uts.edu.au/journals/index.php/provincial_china/article/view/2844

Hu, T., Guo, Q., Shen, X., Sun, H., Wu, R., & Xi, H. (2019). Utilizing Unlabeled Data to Detect Electricity Fraud in AMI: A Semisupervised Deep Learning Approach. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3287–3299. IEEE Transactions on Neural Networks and Learning Systems. https://doi.org/10.1109/TNNLS.2018.2890663

Izonin, I., Tkachenko, R., Shakhovska, N., Ilchyshyn, B., & Singh, K. K. (2022). A Two-Step Data Normalization Approach for Improving Classification Accuracy in the Medical Diagnosis Domain. *Mathematics*, *10*(11), Article 11. https://doi.org/10.3390/math10111942

Jain, Y., Tiwari, N., Dubey, S., & Jain, S. (2019). A comparative analysis of various credit card fraud detection techniques. *International Journal of Recent Technology and Engineering*, *7*(5), 402–407.

Jesus, S., Pombal, J., Alves, D., Cruz, A., Saleiro, P., Ribeiro, R. P., Gama, J., & Bizarro, P. (2022). *Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation* (arXiv:2211.13358). arXiv. https://doi.org/10.48550/arXiv.2211.13358

Jesus, S., Pombal, J., & Saleiro, P. (2022). *Bank Account Fraud Dataset Suite (NeurIPS 2022)* [Dataset]. Kaggle. https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022

Jovanovic, D., Antonijevic, M., Stankovic, M., Zivkovic, M., Tanaskovic, M., & Bacanin, N. (2022). Tuning Machine Learning Models Using a Group Search Firefly Algorithm for Credit Card Fraud Detection. *Mathematics*, *10*(13), Article 13. https://doi.org/10.3390/math10132272

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, *30*. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Khalid, A. R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J., & Adejoh, J. (2024). Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach. *Big Data and Cognitive Computing*, *8*(1), Article 1. https://doi.org/10.3390/bdcc8010006

Khare, N., & Sait, S. Y. (2018). *Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models*.

Kou, Y., Lu, C.-T., Sirwongwattana, S., & Huang, Y.-P. (2004). Survey of fraud detection techniques. *IEEE International Conference on Networking, Sensing and Control, 2004*, *2*, 749-754 Vol.2. https://doi.org/10.1109/ICNSC.2004.1297040

Krishnakumar, A. (2007). *Active Learning Literature Survey*.

Lebichot, B., Paldino, G. M., Siblini, W., He, L., Oblé, F., & Bontempi, G. (2021). Incremental learning strategies for credit cards fraud detection. *International Journal of Data Science and Analytics*, *12*. https://doi.org/10.1007/s41060-021-00258-0

Leite, Miguel Lobo Pinto. (2020). *Active learning for fraud detection* [Universidade do Minho]. https://repositorium.sdum.uminho.pt/handle/1822/84134

Leonard, K. J. (1993). Detecting credit card fraud using expert systems. *Computers & Industrial Engineering*, *25*(1), 103–106. https://doi.org/10.1016/0360-8352(93)90231-L

Li, R., Liu, Z., Ma, Y., Yang, D., & Sun, S. (2023). Internet Financial Fraud Detection Based on Graph Learning. *IEEE Transactions on Computational Social Systems*, *10*(3), 1394–1401. IEEE Transactions on Computational Social Systems. https://doi.org/10.1109/TCSS.2022.3189368

Liu, B., & Tsoumakas, G. (2020). Dealing with class imbalance in classifier chains via random undersampling. *Knowledge-Based Systems*, *192*, 105292. https://doi.org/10.1016/j.knosys.2019.105292

Liu, H., Li, J., & Wong, L. (2002). A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns. *Genome Informatics*, *13*, 51–60.

Liu, Y., Yang, M., Wang, Y., Li, Y., Xiong, T., & Li, A. (2022). Applying machine learning algorithms to predict default probability in the online credit market: Evidence from China. *International Review of Financial Analysis*, *79*, 101971. https://doi.org/10.1016/j.irfa.2021.101971

Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173. https://doi.org/10.1016/j.knosys.2018.10.034

Lopez-Rojas, E. (2016). *Synthetic Financial Datasets For Fraud Detection* [Dataset]. https://www.kaggle.com/datasets/ealaxi/paysim1

Lopez-Rojas, E. (2017). *Synthetic data from a financial payment system* [Dataset]. https://www.kaggle.com/datasets/ealaxi/banksim1

Lopez-Rojas, E. A. (2016). *Applying Simulation to the Problem of Detecting Financial Fraud*. https://urn.kb.se/resolve?urn=urn:nbn:se:bth-12932

Lopez-Rojas, E. A., & Axelsson, S. (2014). BankSim: A Bank Payment Simulation for Fraud Detection Research. In *26th European Modeling and Simulation Symposium, EMSS 2014*.

Lucas, Y., Portier, P.-E., Laporte, L., He, L., Caelen, O., Granitzer, M., & Calabretto, S. (2019). *Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs*.

Maher, P. (2020). *The Seven Most Popular Machine Learning Algorithms for Online Fraud Detection and Their Use in SAS®*.

Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M.-S., & Zeineddine, H. (2019). An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection. *IEEE Access*, *7*, 93010–93022. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2927266

Makower, H., & Marschak, J. (1938). Assets, Prices and Monetary Theory. *Economica*, *5*(19), 261–288. https://doi.org/10.2307/2548629

Melo-Acosta, G. E., Duitama-Muñoz, F., & Arias-Londoño, J. D. (2017). Fraud detection in big data using supervised and semi-supervised learning techniques. *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 1–6. https://doi.org/10.1109/ColComCon.2017.8088206

Merriam-Webster Dictionary, Definition of Fraud. (2024). In *Merriam-Webster Dictionary*. https://www.merriam-webster.com/dictionary/fraud

Mirhashemi, Q. S., Nasiri, N., & Keyvanpour, M. R. (2023). Evaluation of Supervised Machine Learning Algorithms for Credit Card Fraud Detection: A Comparison. *2023 9th International Conference on Web Research (ICWR)*, 247–252. https://doi.org/10.1109/ICWR57742.2023.10139098

Monsen, E. R. (1993). Are you worth your salt? *Journal of the American Dietetic Association*, *93*(2), 136. https://doi.org/10.1016/0002-8223(93)90818-6

Moore, R. C., & DeNero, J. (n.d.). *L1 AND L2 REGULARIZATION FOR MULTICLASS HINGE LOSS MODELS*.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists* (First edition). O'Reilly Media, Inc.

Murugan, K., Felicia, A., Gomathy, B., Saravanakumar, P. T., Ramesh, S. M., & Sakthivel, E. (2023). A Credit Card Fraud Identification Technique Using Support Vector Machine. *2023 International*

Conference on Applied Intelligence and Sustainable Computing (ICAISC), 1–7. https://doi.org/10.1109/ICAISC58445.2023.10199684

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. *Proceedings of the Twenty-First International Conference on Machine Learning*, 78. https://doi.org/10.1145/1015330.1015435

Omair, B., & Alturki, A. (2020). A Systematic Literature Review of Fraud Detection Metrics in Business Processes. *IEEE Access*, *8*, 26893–26903. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2971604

Oxford Advanced Learner's Dictionary, fraud—Definition. (2024, June 22). https://www.oxfordlearnersdictionaries.com/definition/english/fraud?q=fraud

Patel, K. (2023). Credit Card Analytics: A Review of Fraud Detection and Risk Assessment Techniques. *International Journal of Biotech Trends and Technology*, *71*, 69–79. https://doi.org/10.14445/22312803/IJCTT-V71I10P109

Pozzolo, A. D. (2015). *Adaptive Machine Learning for Credit Card Fraud Detection* [PhD]. Université Libre de Bruxelles.

Qi, D., Peng, J., He, Y., & Wang, J. (2023). *Auto-FP: An Experimental Study of Automated Feature Preprocessing for Tabular Data* (arXiv:2310.02540). arXiv. https://doi.org/10.48550/arXiv.2310.02540

Raileanu, L. E., & Stoffel, K. (2004). Theoretical Comparison between the Gini Index and Information Gain Criteria. *Annals of Mathematics and Artificial Intelligence*, *41*(1), 77–93. https://doi.org/10.1023/B:AMAI.0000018580.96245.c6

Ranjon Das, S., Bin Sulaiman, R., & Butt, U. (2023). *Comparative Analysis of Machine Learning Algorithms for Credit Card Fraud Detection*. *1, no. 4, pp. 225–244, 2023*, 225–244.

Rb, A., & Kr, S. K. (2021). Credit card fraud detection using artificial neural network. *Global Transitions Proceedings*, *2*(1), 35–41. https://doi.org/10.1016/j.gltp.2021.01.006

Rtayli, N., & Enneya, N. (2020). Enhanced credit card fraud detection based on SVM-recursive feature elimination and hyper-parameters optimization. *Journal of Information Security and Applications*, *55*, 102596. https://doi.org/10.1016/j.jisa.2020.102596

Russell, S. J., Norvig, P., Chang, M., Devlin, J., Dragan, A., Forsyth, D., Goodfellow, I., Malik, J., Mansinghka, V., Pearl, J., & Wooldridge, M. J. (2022). *Artificial intelligence: A modern approach* (Fourth edition, global edition). Pearson.

Saeed, M. H., & Hama, J. I. (2023). Cardiac disease prediction using AI algorithms with SelectKBest. *Medical & Biological Engineering & Computing*, *61*(12), 3397–3408. https://doi.org/10.1007/s11517-023-02918-8

Saia, R., & Carta, S. (2019). Evaluating the benefits of using proactive transformed-domain-based techniques in fraud detection tasks. *Future Generation Computer Systems*, *93*, 18–32. https://doi.org/10.1016/j.future.2018.10.016

San Miguel Carrasco, R., & Sicilia-Urbán, M.-Á. (2020). Evaluation of Deep Neural Networks for Reduction of Credit Card Fraud Alerts. *IEEE Access*, *8*, 186421–186432. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3026222

Santos, S. J. dos. (2020). *Label Noise Injection Methods for Model Robustness Assessment in Fraud Detection Datasets* [Relatório de Estágio]. Feedzai. https://run.unl.pt/bitstream/10362/112794/1/TAA0073.pdf

Shenvi, P., Samant, N., Kumar, S., & Kulkarni, V. (2019). Credit Card Fraud Detection using Deep Learning. *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 1–5. https://doi.org/10.1109/I2CT45611.2019.9033906

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

Soltani Halvaiee, N., & Akbari, M. K. (2014). A novel model for credit card fraud detection using Artificial Immune Systems. *Applied Soft Computing*, *24*, 40–49. https://doi.org/10.1016/j.asoc.2014.06.042

Soper, D. S. (2023). Hyperparameter Optimization Using Successive Halving with Greedy Cross Validation. *Algorithms*, *16*(1), Article 1. https://doi.org/10.3390/a16010017

Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Mueller, K.-R. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *MACHINE LEARNING AND KNOWLEDGE EXTRACTION*, *3*(2), 392–413. https://doi.org/10.3390/make3020020

Su, Q., Hamed, H. N. A., Isa, M. A., Hao, X., & Dai, X. (2024). A GAN-based Data Augmentation Method for Imbalanced Multi-class Skin Lesion Classification. *IEEE Access*, 1–1. https://doi.org/10.1109/ACCESS.2024.3360215

Sultana, S., Rahman, Md. S., & Afroj, M. (2023). An efficient fraud detection mechanism based on machine learning and blockchain technology. *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 162–168. 2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). https://doi.org/10.1109/3ICT60104.2023.10391306

The New Palgrave Dictionary of Economics. (2008). https://link.springer.com/book/10.1007/978-1-349-58802-2

Thimonier, H., Popineau, F., Rimmel, A., Doan, B.-L., & Daniel, F. (2023). *Comparative Evaluation of Anomaly Detection Methods for Fraud Detection in Online Credit Card Payments* (arXiv:2312.13896). arXiv. http://arxiv.org/abs/2312.13896

Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., & Singh, A. K. (2021). *Credit Card Fraud Detection using Machine Learning: A Study* (arXiv:2108.10005). arXiv. https://doi.org/10.48550/arXiv.2108.10005

Université Libre de Bruxelles, M. L. G. (2018). *Credit Card Fraud Detection* [Dataset]. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Urbanowicz, R., Zhang, R., Cui, Y., & Suri, P. (2023). STREAMLINE: A Simple, Transparent, End-To-End Automated Machine Learning Pipeline Facilitating Data Analysis and Algorithm Comparison. In L. Trujillo, S. M. Winkler, S. Silva, & W. Banzhaf (Eds.), *Genetic Programming Theory and Practice XIX* (pp. 201–231). Springer Nature. https://doi.org/10.1007/978-981-19-8460-0_9

Valverde-Albacete, F. J., & Peláez-Moreno, C. (2014). 100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox. *PLOS ONE*, *9*(1), e84217. https://doi.org/10.1371/journal.pone.0084217

Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2015). APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, *75*, 38–48. https://doi.org/10.1016/j.dss.2015.04.013

Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2017). GOTCHA! Network-Based Fraud Detection for Social Security Fraud. *Management Science*, *63*(9), 3090–3110. https://doi.org/10.1287/mnsc.2016.2489

Vanneschi, L., & Silva, S. (2023). *Lectures on Intelligent Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-031-17922-8

Venter, J., de Waal, A., & Willers, C. (2007). Specializing CRISP-DM for evidence mining. In P. Craiger & S. Shenoi (Eds.), *ADVANCES IN DIGITAL FORENSIC III* (Vol. 242, pp. 303-+). Springer. https://www.webofscience.com/wos/woscc/full-record/WOS:000249667100021

Verleysen, M., & François, D. (2005). The Curse of Dimensionality in Data Mining and Time Series Prediction. In J. Cabestany, A. Prieto, & F. Sandoval (Eds.), *Computational Intelligence and Bioinspired Systems* (pp. 758–770). Springer. https://doi.org/10.1007/11494669_93

Wang, H., Wang, W., Liu, Y., & Alidaee, B. (2022). Integrating Machine Learning Algorithms With Quantum Annealing Solvers for Online Fraud Detection. *IEEE Access*, *10*, 75908–75917. https://doi.org/10.1109/ACCESS.2022.3190897

Wang, Y., Hahn, C., & Sutrave, K. (2016). Mobile payment security, threats, and challenges. *2016 Second International Conference on Mobile and Secure Services (MobiSecServ)*, 1–5. https://doi.org/10.1109/MOBISECSERV.2016.7440226

West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, *57*, 47–66. https://doi.org/10.1016/j.cose.2015.09.005

Wigan, H. (2004). The Effects Of The 1925 Portuguese Bank Note Crisis. *Department of Economic History*.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*.

Wooldridge, J. M. (2019). *Introductory econometrics: A modern approach, 7th ed.* (Seventh edition, student edition). Cengage Learning.

Ye, J., Janardan, R., & Li, Q. (2004). Two-Dimensional Linear Discriminant Analysis. *Advances in Neural Information Processing Systems*, *17*. https://proceedings.neurips.cc/paper_files/paper/2004/hash/86ecfcbc1e9f1ae5ee2d71910877da36-Abstract.html

Yuan, K., Chi, G., Zhou, Y., & Yin, H. (2022). A novel two-stage hybrid default prediction model with k-means clustering and support vector domain description. *Research in International Business and Finance*, *59*, 101536. https://doi.org/10.1016/j.ribaf.2021.101536

Zareapoor, M., & Shamsolmoali, P. (2015). Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier. *Procedia Computer Science*, *48*, 679–685. https://doi.org/10.1016/j.procs.2015.04.201

Zhang, H., Shi, Y., Yang, X., & Zhou, R. (2021). A firefly algorithm modified support vector machine for the credit risk assessment of supply chain finance. *Research in International Business and Finance*, *58*, 101482. https://doi.org/10.1016/j.ribaf.2021.101482

Zhang, X., Han, Y., Xu, W., & Wang, Q. (2021). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. *Information Sciences*, *557*, 302–316. https://doi.org/10.1016/j.ins.2019.05.023

Zhu, H., Liu, G., Zhou, M., Xie, Y., Abusorrah, A., & Kang, Q. (2020). Optimizing Weighted Extreme Learning Machines for imbalanced classification and application to credit card fraud detection. *Neurocomputing*, *407*, 50–62. https://doi.org/10.1016/j.neucom.2020.04.078

APPENDIX A

Default parameters for the models employed in the experimental study

Decision Tree

Parameter name	Default	Comments
criterion	'gini'	Measure quality of split
splitter	'best'	split strategy at each node
max_depth	None	Maximum depth of the tree
min_samples_split	2	min. no. samples to allow for split
min_samples_leaf	1	min. no. samples at each leaf
max_features	None	No. of features to eval for split
max_leaf_nodes	None	Max number of leaf nodes in the tree
min_impurity_decrease	0.0	Threshold to prevent splits that don't improve impurity
class_weight	None	Weights associated with classes
ccp_alpha	0.0	Complexity param (minimal cost-complexity pruning)
monotonic_cst	None	Constraint to relate higher vals of feature with True class

Random Forest

Parameter name	Default	Comments
n_estimators	100	Number of trees in the forest
max_features	sqrt	No. of features to eval for split - default: sqrt of no of cols
bootstrap	True	Use bootstrapping - for train, choose samples with repetition
oob_score	False	Out of bag samples - calculate validation on unused samples
warm_start	False	If True, reuse the solution from the previous call
max_samples	None	If bootstrap=True, number of samp. to draw from train dataset

XGBoost

Parameter name	Default	Comments
objective	binary logistic	
booster	gbtree	
colsample_bylevel	1	
colsample_bynode	1	
colsample_bytree	1	
enable_categorical	False	
gamma	0	
grow_policy	depthwise	
learning_rate	0,3	
max_bin	256	
max_delta_step	0	
max_depth	6	Maximum depth per DT
max_leaves	0	
min_child_weight	1	Minimum sum of weight in a leaf
missing	nan	
multi_strategy	one_output_per_tree	Not relevant for binary classification
num_parallel_tree	1	
reg_alpha	0	L1 regulatization
reg_lambda	1	L2 regularization
sampling_method	uniform	
scale_pos_weight	1	
subsample	1	
tree_method	auto	Faster histogram-opt. approximate greedy algorithm

LightGBM

Parameter name	Default	Comments
boosting_type	gbdt	gbdt=gradient boosting decision tree
num_leaves	31	Maximum tree leaves per DT
max_depth	-1	Maximum depth per DT (-1: no limit)
learning_rate	0,1	Boosting learning rate
n_estimators	100	Number of trees
subsample_for_bin	200.000	Number of samples for binning
objective	None	Learning task objective
class_weight	None	None: do not balance classes
min_split_gain	0,0	Minimum loss reduction for a split to occur
min_child_weight	0,001	Minimum sum of weight in a leaf
min_child_samples	20	Minimum number of data on a leaf
subsample	1,0	Subsample ratio
subsample_freq	0	Subsample frequency. 0=Disable subsamp
colsample_bytree	1,0	Subsample ratio of columns for each DT
reg_alpha	0,0	L1 regularization on weights
reg_lambda	0,0	L2 regularization on weights
n_jobs	None	None means one thread per CPU core
importance_type	split	Criteria to update reported feature importance

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa