N/A

Auteur : Saulas, Adrien
Promoteur(s) : Debruyne, Christophe
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en science des données, à finalité spécialisée
Année académique : 2023-2024
URI/URL : http://hdl.handle.net/2268.2/21035

Université de Liège

Faculté des Sciences Appliquées

Balancing Durability, Performance, and Interpretability in Unbalanced Data as Fraud Detection

ATFE9009-1 - Master thesis

Author:
SAULAS Adrien

Supervisor:
Prof. DEBRUYNE Christophe

Intech Advisor:
NOGATCHEWSKY Matthieu

August 21, 2024

Acknowledgments

I would like to express my sincere gratitude to my industrial supervisor, Matthieu Nogatchewsky, for his guidance and support throughout this study, as well as for his high availability and patience. I would also like to thank my academic supervisor, Christophe Debruyne, for his valuable advice and assistance in the writing of this thesis.

I would also like to thank my colleagues at Intech, whose conversations over coffee helped move this project forward. In particular, I would like to thank Alexandre, Jacques, Fred, Mathieu, Silvio, Guillaume, Kimberley, Ibrahim, Serge and Anthony, who made this internship at Intech an extraordinary experience.

I would also like to express my gratitude to my family, who have always supported me throughout my studies and this thesis.

I extend my thanks to my friends, Lalie, Emilie, Adrien, and the C.A.Ps, without whom these years would not have been the same.

Finally, I want to give a special thanks to my best friend and third brother, Thomas Gillet, for his support (both technical and ethical) and for being there during the more challenging moments.

Abstract

The problem of fraud detection is one of the most discussed topics in the field of machine learning. This study addresses four key areas essential for a fraud detection platform: prediction accuracy in imbalanced datasets, interpretability of predictions, deployment and sustainability of the platform, monetary costs associated with model errors. To tackle these issues, we first conducted extensive research in the field, then proposed and evaluated our solutions. We introduce methods such as using a WCGAN (Wasserstein Conditional Generative Adversarial Network) for sampling or cost-sensitive learning with new models like Light Gradient Boosting, employing interpretable models like Explainable Boosting, deploying and automating training processes with Kubernetes and Kubeflow, and utilizing approaches like thresholding or tuning metrics that account for monetary costs. Each of these solutions shows promising results and improves upon existing research in the field.

Contents

List of Figures

1 Introduction

1.1 Problem formalisation

The Oxford Dictionary defined fraud as "*a type of dishonesty calculated to obtain advantage, generally financial advantage, by some wrongful means (a tort or crime)*"[1]. In the case of preventing these bank transfer frauds, fraud detection and fraud prevention are two essential tools to minimize the losses caused by them. Fraud prevention involves proactive strategies aimed at thwarting fraud before it occurs, whereas fraud detection systems are activated when criminals manage to bypass fraud protection measures and carry out a fraudulent transaction.

Since the advent of credit cards, various types of fraud have emerged over the years. From simple credit card theft to more sophisticated techniques facilitated by the use of the internet. In fact, fraudulent activities on the Internet are rapidly increasing due to the global accessibility of the web and the ease with which users can conceal their identity and location during online transactions.

Currently, numerous preventive fraud security measures have been implemented to protect against bank transfer fraud, such as PIN codes, NRI (Never Received Issue), beneficiary verification, strong authentication, or identification to prevent interception of credit cards before their initial use. However, these numerous preventive measures are currently not sufficient to contain the expansion of bank transfer fraud[2].

As a result, it is important to focus on the concept of fraud detection in order to promptly identify these bank frauds and pursue necessary legal actions. The principle of fraud detection can be divided into two main parts:

1. Firstly, the detection of these frauds based on predefined rules. This approach involves defining a set of rules or specific criteria to identify transactions that may be fraudulent. Considered criteria may include the location of the transfer, similarity with the user's transaction history, the amount and frequency of transfers, the website where the transfer is made, etc. While these predefined rules are effective, they can be circumvented once known.

2. Secondly, fraud detection through machine learning on historical bank transfer data. The advantage of these models is that they can be constantly updated to remain effective and cannot be easily circumvented. However, according to Intech, they face numerous challenges such as data protection, the low occurrence of fraud in the dataset, the possibility of making errors leading to significant costs or loss of clients, and the complexity of the models which can lead to difficulties in interpretation by humans.

Since machine learning is a continuously evolving science, numerous new technologies haven't been explored. Therefore, in this project, we will attempt to address the challenges related to the field of machine learning and try to find the most suitable solution to the problem of fraud detection.

[1] Oxford Dictionary, last checked 11/07/2024 on https://www.oxfordreference.com/

[2] Synetics Solution, The evolution of fraud prevention, detection and risk mitigation, 18 may 2017, last checked: 11/07/2024 on https://www.synectics-solutions.com/our-thinking/evolution-of-fraud-prevention-detection-and-risk-mitigation

1.2 Challenges

According to Intech and the literature, implementing a fraud detection platform using machine learning involves four major challenges that we have attempted to address in this thesis.

1.2.1 Model Development in Imbalanced Situations

As will be further explained in 1.5, fraud detection is particularly complex due to the imbalance in the data. This imbalance significantly affects the results of machine learning models. Several solutions exist to address this issue, such as sampling, cost-sensitive learning, or the use of ensemble models.

1.2.2 Model Development in the Context of Interpretability

The field of fraud detection is particularly complex because each decision can have significant consequences. Therefore, it is crucial to obtain explanations for the decisions made by the models. Fraud detection without explanation cannot lead to real judicial proceedings. This is why, in this thesis, we will conduct research on interpretable models and explainable methods.

1.2.3 Continuous Training and Application Deployment

Many studies in the field of fraud detection have already been conducted in the past. So why is it still interesting to pursue such a research project today? As explained in this article[3], many machine learning models are not industrialized and accessible through a real platform. Additionally, the training of these machine learning models is often frozen in time, relying on historical data. To achieve a truly sustainable product, it is necessary to approach the concepts of application deployment and continuous model training over time through monitoring by fraud experts. Therefore, in this thesis, we will attempt to address these concepts and create a method that allows for the continuous training of models over time.

1.2.4 Approach Related to Monetary Costs

The final challenge addressed in this thesis is the monetary value associated with errors in fraud detection. Indeed, a false accusation can sometimes result in costs of hundreds of thousands of euros. However, these costs depend heavily on the information we have about them (whether we know the direct cost of an error or not), the time available for model retraining based on the cost, and the performance we desire. Based on these three factors, we will aim to develop three concrete solutions for each problem.

[3]Machine Learning: How to Industrialize Your Models? onepoint, last checked 11/08/2024 on https://www.groupeonepoint.com/fr/publications/machine-learning-comment-industrialiser-vos-modeles/

1.3 Mathematical formulation of the problem

Let $T = \{t_1, t_2, ..., t_n\}$ be a set of transactions, where each transaction t_i is represented by a feature vector $x_i \in \mathbb{R}^d$. The goal is to build a classification function $f : \mathbb{R}^n \to \{0, 1\}$ that assigns a label $y_i \in \{0, 1\}$ to each transaction t_i, where $y_i = 0$ indicates a legitimate transaction and $y_i = 1$ indicates a fraudulent transaction.

In practice, we approximate the function f using machine learning techniques, such as logistic regression, random forests, gradient boosting, or deep neural networks. These techniques allow us to learn a function f^* from a labeled dataset $D = \{(x_1, y_1),$ $(x_2, y_2), ..., (x_m, y_m)\}$ where m is the number of labeled transactions. Our goal is to obtain a function f^* that is as close as possible to the true function f. In fact, if $D \subset \mathbb{R}^{(n+1)\times m}$ and under certain assumptions, it can indeed be demonstrated that by minimizing the loss function, we can find an approximate function f^* that closely approximates the true function f. The complete demonstration can be found on my introduction to machine learning course [1].

We can show that as the size of the representative dataset D increases, the difference between the learned function f^* and the true function f will converge to zero in probability. In other words, with a large enough dataset, we can find an approximate function f^* that is very close to the true function f with high probability.

1.4 Company Description

This thesis was carried out in collaboration with Intech.

Founded in 1995, InTech is a Luxembourgish IT Services and Engineering Company employing 130 specialists in the design and implementation of business software solutions, including specific developments and integration of generic industrial components. A human-sized company, Intech primarily operates in project mode, combining multidisciplinary skills in project management, architecture consulting, technical expertise, and development. On May 6, 2014, InTech became a subsidiary of the POST Luxembourg group. This decision solidifies InTech's position as a leading digital services company in Luxembourg.

1.4.1 Training

During this internship with Intech, I had the opportunity to participate in numerous training sessions to better understand the skills necessary for a consulting engineer. I attended training on agile methods, CI/CD and testing, front-end and back-end development, and deployment using Docker and Kubernetes. These training sessions took place over 12 days of my internship. It is important to note that these 12 days were in addition to the 80 days allocated for my thesis, in accordance with the requirements for completing a thesis in a company setting.

1.4.2 Work in the Company and Team Building

During this internship, I also had the chance to participate in various team-building activities, such as the LuxiO Day, a conference day organized within Intech for its employees. Additionally, some team-building events among interns helped foster a positive atmosphere within the company among different colleagues.

1.5 Datasets

In this thesis, several datasets needed to be considered to best develop the fraud detection platform.

1.5.1 Post Group Luxembourg Dataset

This thesis was initially in collaboration with the Post Luxembourg group, of which Intech S.A. is a subsidiary. We were supposed to obtain a dataset of historical transactions involving Post Luxembourg members. However, due to GDPR issues and other internal problems at Post, Intech was unable to access this dataset.

As a result, we (along with Intech) decided to use a public dataset[1]. This choice not only allows for the verification of our research results but also enables us to potentially market this platform to other clients.

1.5.2 European Transactions by ULB

After searching for public datasets, the one provided by the Université Libre de Bruxelles (ULB) [1] seemed to be the most used in the field of fraud detection. This dataset contains credit card transactions made in September 2013 by European cardholders. It includes transactions that occurred over two days, with 492 frauds out of a total of 284,807 transactions. The dataset is highly imbalanced, with the positive class (frauds) representing 0.172% of all transactions.

This dataset was then fully anonymized using PCA transformation. Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a dataset into a set of orthogonal components ordered by the amount of variance they explain. Here the variables $V_1, V_2, \ldots, V_{28}$ represent the 28 first principal components of these various variables. Only two variables were not transformed: **Time**, which represents the time in seconds since the first recorded transaction, and **Amount**, which represents the amount of the transaction. Finally, the variable **Class** represents the label of the transaction, being 1 if the transaction is fraudulent and 0 otherwise.

[1]Credit Card Fraud Detection Dataset, provided by ULB on kaggle, last checked 11/07/2024, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

2 Related Work

As previously stated, the purpose of this thesis is to find a representation of a fraud prediction model in the context of bank transfers as well as credit card payments. During my research, we reviewed several papers related to the broader field of fraud detection in bank transfers.

In this review of related work, we will first consider various research efforts conducted in this field, organizing these papers into four main parts:

1. Firstly, the formulation of the problem and the creation of the Fraud Detection System Design principle.
2. Next, the research undertaken to address the issue of class imbalance, which is one of the main challenges in fraud detection.
3. Subsequently, an evaluation of the various models used and the results they have achieved in the scope of fraud detection.
4. Finally, we will consider the research conducted in the area of interpretable models, which appear to be underutilized in the field of fraud detection.

2.1 Different Fraud management possibilities

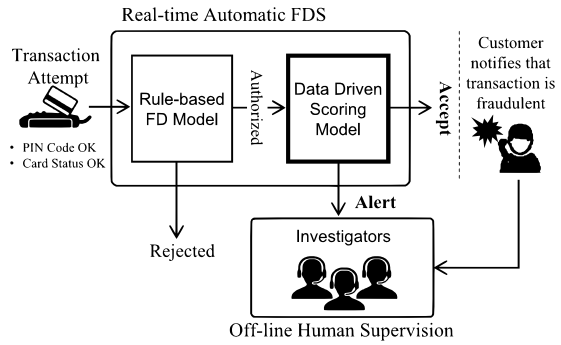
In this section, we will consider how the principle of fraud detection has been defined and what process has been considered to identify to find solution to this problem.

Firstly, the field refers to the solution to the fraud problem as Fraud Detection System Design. This well-known concept today was primarily introduced by Andrea Dal Pozzolo, and Olivier Caelen [2]. The basic principle upon which this concept was formed is succinctly and comprehensively explained by Eunji Kim [3], as illustrated in Figure 1a. In this concept, we can observe two parts: the first representing a set of precises if-then rules designed to highlight certain predefined characteristics studied in advance by specialists. Then, the Data-Driven Scoring Model represents the machine learning aspect aiming to alert the specialist to potential fraudulent transactions.

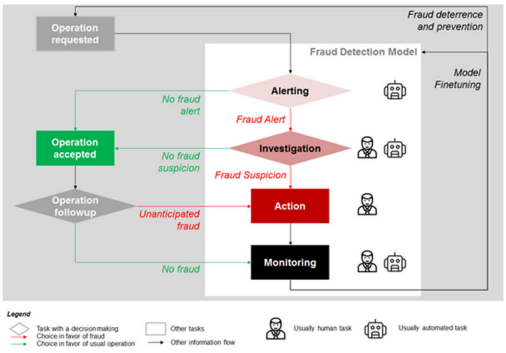
Subsequently, Wesley Kenneth Wilhelm [4] proposed a method dividing the part managed by the AI model and the experts into four parts. These four parts were synthesized and explained in " The accuracy versus interpretability trade-off in fraud detection mode" [5], and their approach can be visualized in Figure 1b. In this approach, they propose the idea of monitoring and fine-tuning the labels to continually improve the models and allow them to adapt to changes in the fraud domain, such as the emergence of new fraud types or shifts in data distribution (changes in consumer interests, etc.).

Next, Mr Dal Pozzolo [2] proposes a scoring system rather than a binary alert system. This approach was developed following an analysis of specialists' problems. Indeed, specialists often did not have the time to analyze all frauds detected by a model. To address this issue without disregarding fraud cases, they devised a scoring system ranging from 1 to 5. Initially, this system was manually labeled, and later, it was labeled directly by the model monitored by a specialist (an idea proposed by Eunji Kim [3]). This scoring system rates transactions from one star, indicating almost certainly not fraud, to five stars, indicating definitely fraud. This scoring system enables specialists to focus on transactions with higher scores to avoid wasting time on potentially legitimate transactions. The FDS envisioned at this point by Mr Dal Pozzolo [2] can be found in Figure 1c.

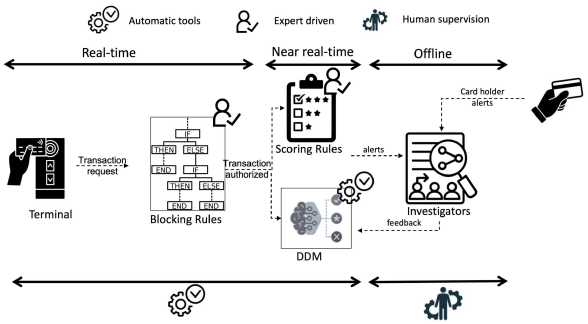
Finally, Mr Dal Pozzolo [6] improve his FDS model by introduced the idea of adding a model to assess the risk that a transaction is fraudulent before applying precise blocking rules as expressed by in the figure 1d. This allowed for the implementation of more time-consuming rules if the risk was moderately elevated and also to block certain transactions earlier when the risk was too high. Then if the risk is low or moderate the transaction is passing to a data-driven scoring model called 3DDSM, which is compute by using the last technologies of cloud computing.



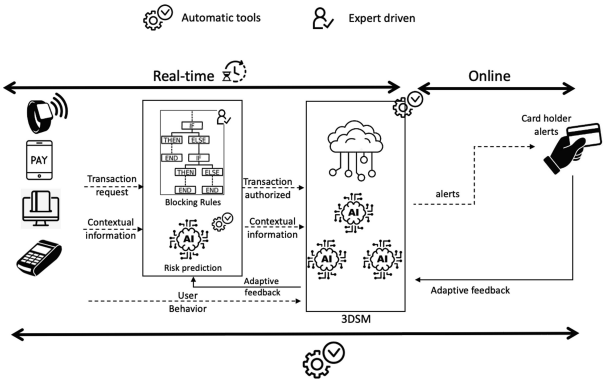
(a) Fraud detection system design an overview, [3]



(b) FDS with division of the model process, [5]



(c) FDS with scoring methods, [2]



(d) Last version of FDS [6]

Figure 1: Evolution of Fraud detection design

A new possibility that is widely discussed in the field of fraud detection is to no longer consider frauds as a transaction with associated metadata, thus stored in a database in tabular form, but to visualize the transactions as data exchanges between two people. This data can be stored in databases in the form of graphs such that each account is a node and each transaction represents an edge between two nodes. This new approach is due to the emergence of Graph Neural Networks, which are proving to be increasingly effective. An example of a GNN through message passing applied to fraud detection and providing good results had a performance over 95% in F1 score [7]. The major advancement they made is to manage to include information from unlabeled nodes in the GNN, which provides context to a labeled node based on its neighbors.

Some other projects have tried other strategies such as using attention models with LSTM [8] or the use of transformers adapted for fraud detection [9], however, these different research efforts have not brought significant advances in the field.

In this project, we will develop an application to alert, investigate, and monitor bank

transfers. Due to InTech's constraints, we have a binary dataset rather than a dataset labeled with scores. Additionally, given the various challenges to be addressed and the time allocated for this research, we will not explore the option of adding a machine learning model in the initial steps.

2.2 Imbalenced management

Imbalanced class problems are a major issue in fraud detection. The principle of imbalanced classes simply refers to the situation where one class (here, fraudulent transactions) is much less prevalent than another class (legitimate transactions). Several solutions exist to best manage these problems.

- **Oversampling:** the practice of synthetically generating data in the minority class.
- **Undersampling:** the practice of removing data from the majority class while trying to minimize information loss.
- **Cost-sensitive learning:** adjusts the model weights to address the unequal importance and frequency of classes by assigning higher costs to errors involving the minority class.

2.2.1 Oversampling

Most fraud detection projects start by using oversampling as the first method to handle imbalanced class issues, typically employing techniques such as SMOTE and its variants (ADASYN, BorderlineSMOTE, etc.), all of which are provided by imbalanced-learn [10]. Although these methods are functional in some models, they have their limitations. Indeed, these methods do not always result in performance improvements, and other methods such as the cost-sensitive approach can outperform them. We will use these oversampling methods as a fundamental approach to see if the new methods discussed perform better.

Firstly, Peter Gnip [11] introduces the Selective Oversampling Approach (SOA). SOA's strategy involves two steps. It starts with a one-class support vector machine (OCSVM) trained on the majority class to detect outliers in the minority class. This helps identify minority samples that might mislead the classifier due to atypical characteristics. These misclassified samples are removed, and only those that are correctly identified and representative of their class are retained. SOA then balances the training dataset by generating synthetic samples from these retained minority class observations using SMOTE or ADASYN. A schema of the SOA models can be found on 2a. However, while this method seems relevant and useful within the scope of this project, the code is no longer accessible and therefore we will no more investigate this approach.

Subsequently, Justin Engelmann [12] proposes an approach with a Generative Adversarial Network named cWGAN-based. A GAN consists of a generator, which creates data, and a discriminator, which evaluates their authenticity. The generator learns to produce increasingly realistic data, while the discriminator improves in detecting fakes, thus creating a kind of competition that enhances the quality of the generated outcomes. The model proposed adds some key concepts that lead to real improvements.

1. **Effective management of mixed data types:** cWGAN-based is capable of simultaneously processing numerical and categorical data in tabular form, essential

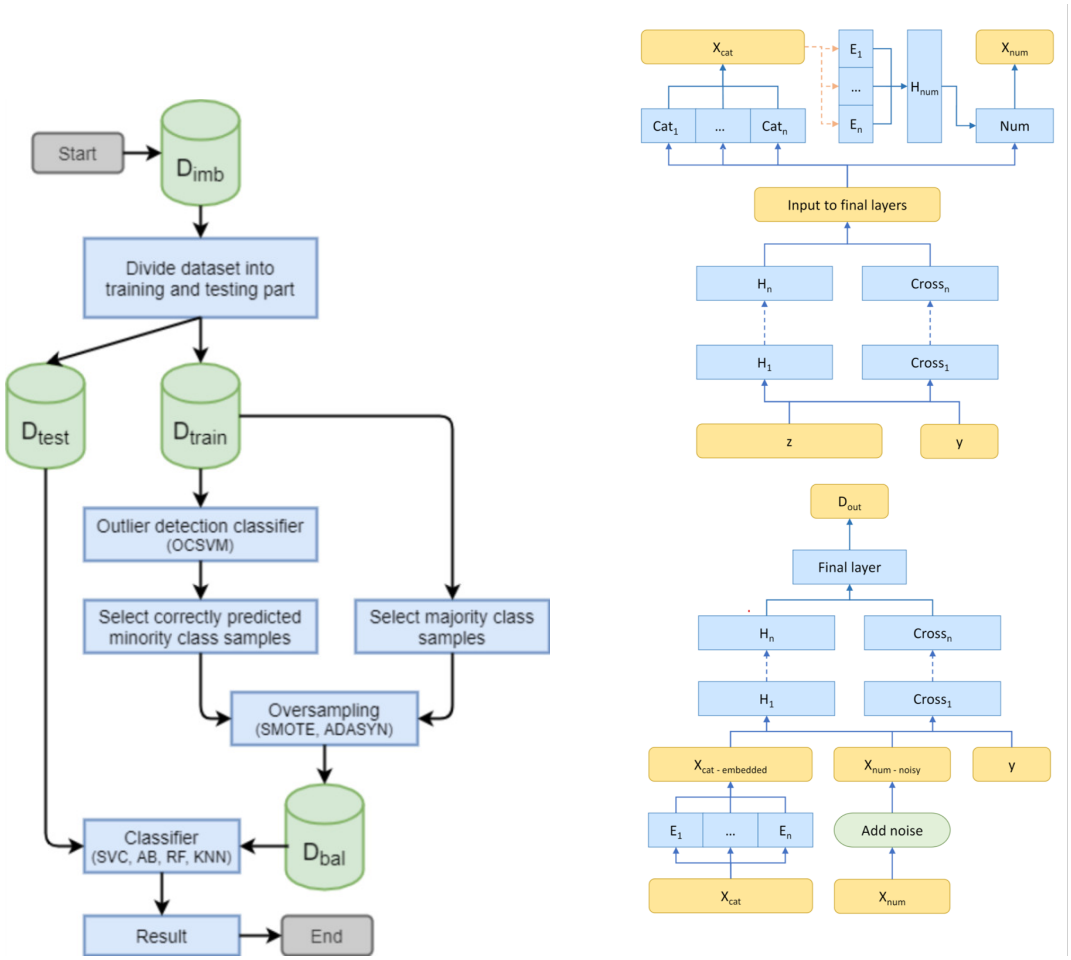
for applications such as credit scoring which often involve mixed data types.

2. **Auxiliary classifier loss:** cWGAN-based incorporates an auxiliary classifier in the GAN training process, ensuring that the generated samples are not only realistic but also relevant for downstream classification tasks. This is achieved by adding an auxiliary classifier loss to the GAN training objectives.

3. **cWGAN framework:** Use of the Wasserstein loss with a gradient penalty (WGAN-GP) to generate high-quality samples by stabilizing the GAN training process.

4. **Conditionally train the generator** The generator G now estimates a conditional probability $P(X|y)$ and the discriminator estimates $D(X, y) = P(\text{fake}|X, y)$, which is the probability that the sample of a specific class is not real. In this way, we can generate synthetic individuals of the minority class by considering the distribution of the data.

The structure of cWGAN-based can be found in 2b. At the top of the figure, the generator schema, where 'Cross' corresponds to the Crosslayers wich calculate feature interaction layer, and 'Cat' are the parts associated with categorical data. Similarly, the bottom part of the figure shows the schema of the discriminator.



(a) Selective Oversampling Approach models [11]

(b) Generator and Discriminator Structure of the WCGAN [12]

Figure 2: Elaborate oversampling methods schema

Finally, Zhuoyuan Zhen [13] create a method called Sigma Nearest Oversampling based on Convex Combination (SNOCC), designed to address the limitations of SMOTE in handling imbalanced datasets. The method is developed to produce synthetic samples that better reflect the original distribution of minority class samples.

SNOCC incorporates several key improvements:

1. **Convex Combination of Seed Samples:** Unlike SMOTE, which linearly interpolates between pairs of seed samples, SNOCC uses a convex combination of multiple seed samples to generate synthetic instances. This allows for a more diverse representation of the minority class in the feature space.
2. **Advanced Nearest Neighbors Search:** SNOCC employs an enhanced algorithm for nearest neighbor search, which not only looks at the closest neighbors but considers a broader range based on a dynamically calculated sigma value. This prevents the overlapping of classes and produces a more dispersed sample distribution.
3. **Cluster-Based Approach:** Prior to oversampling, SNOCC may utilize clustering techniques to group similar instances within the minority class, enhancing the relevancy and variety of the synthetic samples generated.

However, while this method seems relevant and useful within the scope of this project, the code is no longer accessible and therefore we will no more investigate this approach.

2.2.2 Undersampling

Undersampling is a commonly used technique to address the problem of imbalanced dataset by balancing the classes through the reduction of instances from the majority class. This method can be implemented in three principal manners:

1. **Pure Undersampling**: This approach directly reduces the number of majority class instances to balance the dataset. Key techniques include:
   - **Condensed Nearest Neighbor (CNN)**:

     CNN [14] is an early prototype selection method that aims to reduce the training dataset by keeping only those samples that are necessary to retain the classifier's performance. It selects a subset of points that can classify the remaining points with the same accuracy as would the entire set.
   - **Tomek Links**:

     The Tomek links [15] method identifies pairs of very close instances, but of opposite classes. Removing such Tomek links can help in cleaning overlapping data points and thus can improve the decision boundary between classes.
2. **Hybrid Undersampling**: This strategy combines undersampling with other data processing techniques to enhance the effectiveness of the undersampling process, such as:
   - **Cluster-based Undersampling**:

     Cluster-based Undersampling [16] involves clustering the majority class instances and then performing undersampling within each cluster to ensure that

the undersampling process does not affect the representativeness of the majority class. It leverages the internal structure of the data to maintain diversity within the sampled dataset. Currently, the most commonly used algorithm that employs this clustering concept is Centroid Clustering, provided by the imbalanced-learn library [10].

3. **Undersampling with ensemble or evolutionary methods**: These methods integrate undersampling into ensemble learning frameworks or use evolutionary algorithms to optimize the subset selection, such as:

   - **RUSBoost**:

     RUSBoost [17] is an adaptation of the AdaBoost algorithm which incorporates random undersampling. It specifically addresses class imbalance by iteratively focusing more on incorrectly classified minority samples.

While some other undersampling techniques, which seem more elaborate, have been published [18]. We retained the algorithms and the research that could be used in this project or where the code was open source.

To summarize, undersampling helps to improve model performance by preventing bias towards the majority class, however it is crucial to apply it judiciously to avoid significant information loss.

2.2.3 Cost Sensitive Learning

The idea behind cost-sensitive learning (CSL) is to integrate the concept of asymmetric error costs directly into the machine learning algorithms' learning process. In practice, not all classification errors are equal in terms of consequences: some may have more severe repercussions than others. Therefore, CSL aims to minimize not only the number of errors but, more importantly, the total cost associated with these errors.

Much research has been conducted in the field of cost-sensitive learning, which can be classified into four main methods:

1. **Relabeling:**

   As part of the preprocessing, one of the best-known methods is MetaCost [19]. In this algorithm, a model is placed upstream of training our model to adjust the actual values of the classes in order to minimize costs. The advantage of this method is that it can be applied to any model and is easily implementable. However, the training time is therefore doubled, and it heavily depends on the predictions of the base model.

2. **Instance Weighting:**

   Instance weighting introduced by Kai Ming Ting [20] and improved [21] over the years, is a technique that assigns different importances to examples in a dataset, guiding the learning algorithm to focus more on specific examples. This method allows adaptability to each instance if desired, but if misused, can quickly lead to overfitting of our model.

3. **Direct Approach:**

In this approach, the costs associated with classification errors are directly integrated into the learning algorithm itself. This may involve modifying loss functions, split criteria in decision trees, or other internal elements of the model. Generally, this method allows for better performance on specific cost criteria. In this method, costs are intrinsically linked to the model and thus directly depend on the algorithm used. An example of a direct approach was developed by Charles X. Ling [22] to apply a cost-sensitive approach to decision trees.

4. **Thresholding:**

   Thresholding [23] involves adjusting the decision threshold of a classifier to account for different costs. By default, many classifiers use 0.5 as the threshold in binary classification tasks, but this threshold can be modified to reduce the impact of costly errors. Thresholding is easy to implement and can be done without needing to retrain the model. Certain thresholds can minimize overall costs when adjusted based on them.

2.3 Models Used in Fraud Detection

This section is dedicated to the different results of models or preprocessing algorithms other than those dealing only with data imbalance, which have achieved good results. In order to conduct a concrete analysis with my work later, we searched for research papers providing results on the same public dataset that we used. The different result can be found on Table 1. We can already see that we outperform all result in the field with our different approach evaluate in Section 4

2.3.1 Preprocessing

1. **Genetic Algorithm for feature selection** [24]

   In this paper they use the Genetic Algorithm for feature selection. The idea behind this approach follows these steps:

   (a) Create subsets of features and evaluate the fitness of these subsets with a classifier (Random Forest in this case).

   (b) Select the best-performing feature subsets based on their fitness scores.

   (c) Combine pairs of feature subsets from the mating pool to produce new offspring.

   (d) Introduce random changes to some feature subsets to maintain diversity and explore new potential solutions.

   (e) Form a new population by replacing some or all of the old population with the new offspring.

2. **Pipelining and ensemble methods** [25]

   The idea proposed here is to use a pipeline. Pipelining involves applying a series of transformations followed by a final classifier. In this work, selectKBest from sklearn is used to select features based on the top scores, utilizing f-regression for univariate linear regression tests to determine the influence of each feature. Finally, a Random Forest Classifier is used for classification and prediction.

2.3.2 Modelling

1. **Boosting model**

   In this paper [26] they conducted a comparative study between many models as well as numerous sampling algorithms. In their research, they noticed that Boosting algorithms such as CatBoost, Extreme Gradient Boosting, and Light Gradient Boosting provided better results than the traditionally used ensemble models. Furthermore, undersampling methods such as A11 K-Nearest-Neighbors and oversampling methods such as SVMSmote appear to be the most appropriate methods for Boosting models.

Some other research papers [27], had new ideas when tackling the problem of fraud detection. However, it seems that the sampling was done on the whole dataset, which created a lot of overfitting or errors in the calculation of the metrics.

Table 1: Table representing the result of different models with their prepossessing, ? noticing a maetric value not provided in the search paper.

Model	Accuracy	Precison	Recall	F1	AUC
GA-Random-Forest-full	0.87	0.926	0.777	0.846	?
GA-Random-Forest-v5	0.998	0.725	0.953	0.824	0.95
GA-Random-Forest-v4	0.999	0.778	0.838	0.807	0.95
Pipelining	0.999	0.84	0.86	0.85	?
CatBoost-A11KNN	0.999	0.803	0.959	0.874	0.979
CatBoost-OSS	0.999	0.795	0.961	0.86	0.981
XGB-OSS	0.999	0.7947	0.9541	0.8669	0.976
XGB-Border-Smote1	0.999	0.823	0.929	0.873	0.964
RF-SVM-Smote	0.999	0.792	0.953	0.865	0.976
Repeated Edited NN	0.999	0.788	0.958	0.865	0.978

2.4 Interpretability and Explainability

The explainability of machine learning models in fraud detection is crucial for building user trust and ensuring fair decision-making. As fraud detection systems become increasingly complex, understanding how and why certain decisions are made becomes essential and so for three reason according to Intech.

1. **Scientific Perspective:** It is important to understand how the model thinks to better comprehend the studied phenomenon and thus verify the coherence of the results.
2. **Ethical Perspective:** If the platform refuses an individual's transaction without explanation and therefore without justification by an expert afterward, it is unacceptable.
3. **Legislative Perspective:** Article 22 of the GDPR states: *"The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly*

significantly affects him or her."[4]

This involves demystifying the internal mechanisms of algorithmic models to ensure their transparency and accountability. Indeed, by providing understandable explanations of model outcomes, institutions can not only justify the actions taken but also enable experts to regulate and continuously improve these systems. Explainability thus becomes an indispensable bridge between the technical capabilities of fraud detection models and their social and regulatory acceptability.

2.4.1 Explainability vs Interpretability

Explainability and interpretability are closely related yet distinct concepts. Interpretability refers to ***"the ability of a model to present its decisions in a comprehensible manner to a human,"*** [28], thus enabling the tracing of cause-and-effect relationships between system inputs and outputs. It focuses on the intuition behind a model's results. In contrast explainability can be defined as ***"the understanding of the internal mechanisms and operational logic of a machine learning system, aiming to reveal the internal processes during the model's training or decision-making"*** [29]. These two dimensions, although complementary, do not guarantee each other, making the integration of both aspects crucial for enhancing the transparency and trust in machine learning systems.

2.4.2 The Scope of Interpretability and Explainability

Pantelis Linardatos [30] proposes a classification of interpretable models based on four main characteristics, providing a better understanding of how models can be explained and analyzed, as shown in Figure 3.

1. **Local vs Global**:
    - **Local**: Explains individual predictions, detailing why specific decisions were made for given samples.
    - **Global**: Provides an overview of a model's logic, enabling an understanding of its overall operation.
2. **Data Types**:
    - The different models can be divided based on the types of data they can handle, such as tabular data, text, images, graphs, etc.
3. **Purposes of Interpretability**:
    - Includes objectives such as creating inherently transparent models, explaining complex models after their development, enhancing model fairness, and evaluating the sensitivity of predictions.
4. **Model Specific vs Model Agnostic**:
    - **Model Specific**: Methods designed for particular models.
    - **Model Agnostic**: Techniques applicable regardless of the model's architecture, offering increased flexibility.

[4] Article 22 of the GDPR, last checked 11/08/24 on https://www.gdpr.org/regulation/article-22.html

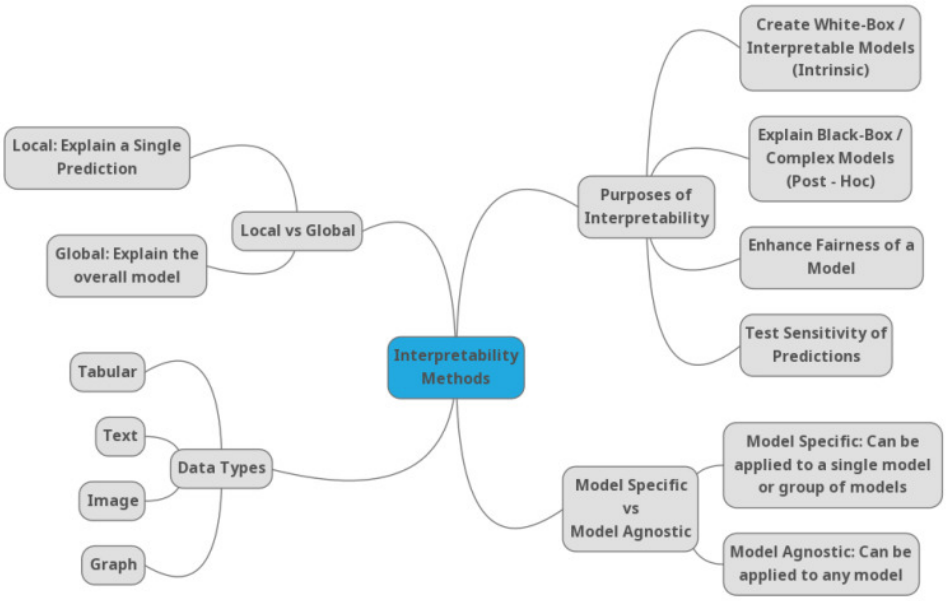


Figure 3: Mind-map of the different particularities of interpretable models [30]

In our case, we will focus on locally and globally interpretable models since we want as much interpretability as possible, whether they are agnostic or specific, to tackle as many methods as possible for tabular data given our dataset. Our aim is to either create interpretable white-box models or explain black-box models to both explain the most performant models (black-box) and have models where we can fully explain the predictions (white-box).

2.4.3 White-box Models

White-box models in machine learning are characterized by their transparency and inherent interpretability. They are designed so that users can clearly understand how inputs are transformed into outputs. Traditional examples of white-box models include simple decision trees and logistic regression. However, recent advancements have highlighted two particularly complex approaches that maintain clarity and usability:

1. **Supersparse Linear Integer Models (SLIM)**

   SLIMs [31] utilize basic arithmetic operations such as addition, subtraction, and multiplication on input features. This method enhances interpretability by employing straightforward computational processes, making it easier to comprehend how inputs are transformed into outputs. Despite being open-source, available on GitHub, there has been little recent activity on its development, and the model has seen limited use in real-world contexts.

2. **Generalized Additive Models with Pairwise Interactions (GA2Ms)**

   Before introducing GA2M we need to introduce the principle of generalized additive models (GAM).

   Generalized Additive Models [5] (GAM) are a type of statistical model that combines

[5]GAM: The Predictive Modeling Silver Bullet, KIM LARSEN, 30/07/2015, last checked 11/07/2024

the properties of generalized linear models (GLM) with additive models. The key idea is to model the response variable as a sum of smooth functions of the predictors. Each predictor can have its own smooth function, which can capture nonlinear relationships.

Mathematically, a GAM can be represented as:

$$g(E[y]) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

or

$$g(E[y]) = \beta_0 + \sum f_j(x_j)$$

where:

- β_0 is the intercept,
- $f_i(x_i)$ are smooth functions of the predictors x_i

The smooth functions f_i are typically estimated using techniques such as splines or kernel smoothing, allowing for flexible and interpretable modeling of the data. GAMs are particularly useful when the relationship between the predictors and the response is not strictly linear, providing a balance between flexibility and interpretability.

Lets introduce GA2M [32]. They propose to improve the GAM by incorporating modern machine learning techniques such as bagging and boosting. They also automatically include pairwise interactions, which significantly boost their predictive power. Such that the mathematical expression of GA2M is

$$g(E[y]) = \beta_0 + \sum f_i(x_i) + \sum f_{i,j}(x_i, x_j)$$

Where:

- β_0 is the intercept.
- $\sum f_i(x_i)$ represents the sum of functions f_i applied to each predictor x_i.
- $\sum f_{i,j}(x_i, x_j)$ denotes the sum of interaction terms between pairs of predictors, modeled by functions $f_{i,j}$ also called FAST function in the paper [32].

The pairwise interaction is calculated with a method introduced in the paper called FAST. Instead of building a full interaction model, which is computationally expensive, FAST estimates the benefit of modeling interactions by reducing the Residual Sum of Squares (RSS). It should also be noted that for large datasets (which is our case here), in order to reduce computation time, FAST retains only the K best interactions for training, K is chosen according to computing power

The structure of GA2Ms allows for a clear calculation and understanding of each feature's contribution, enhancing both their interpretability and effectiveness. Contrary to SLIM, GA2M has been implemented in the Explainable Boosting Machine model, which is part of the InterpretML library. This model is easily importable and added to a project. This model will be more detailled in the Modelling part.

on https://www.taylorfrancis.com/books/mono/10.1201/9781315370279/generalized-additive-models-simon-wood

These models exemplify significant strides in the field of interpretable machine learning, illustrating the potential to combine high accuracy with transparent, understandable decision-making processes.

2.4.4 Black box Model

This second category includes methods that aim to make pre-trained models explainable. These models are called black-box types because they are naturally difficult to interpret and explain. In summary, these methods do not attempt to create interpretable models, but rather to explain already trained models.

There are many algorithms for the interpretability of black-box models; however, two seem to stand out and be more widely used.

1. **SHAP (SHapley Additive exPlanations)**

   Before introducing SHAP, it is necessary to introduce Shapley values [33], which derive from cooperative game theory and evaluate the contributions of individual players (features, in this context) to the overall game (the prediction result). To simplify this concept, let's assume there are three variables in a prediction (V1, V2, V3) as shown in Figure 4. We will look for the Shapley value of V1. To do this, we consider all possible combinations of variables with and without V1 and calculate the probability of being a fraud for each case. By averaging these probabilities, we obtain the Shapley value of this variable.

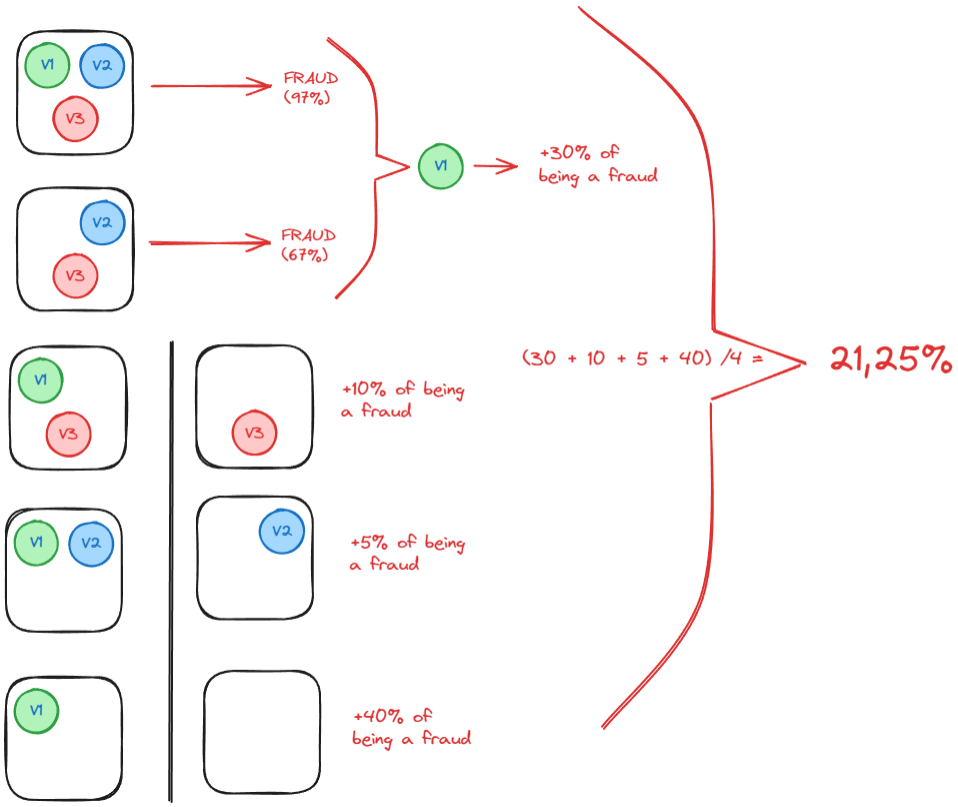


Figure 4: Shapley values calculation

The advantage of this method is that it is based on clear and understandable theo-

retical concepts, which allows for a real understanding of the interpretability results we obtain. However, the complexity of calculating a Shapley value is $O(N!)$, which can be problematic when the number of features is large (as is the case this thesis).

SHapley Additive exPlanations (SHAP) [34] intervenes to overcome the limitations of Shapley values. SHAP creates various approximations to optimize the calculation of Shapley values. However, for these approximations to be valid, they must adhere to the three principles introduced by SHAP:

(a) **Accuracy**
The explanation model should accurately represent the original model's output for a specific input instance. In other words, the sum of the SHAP values for all features plus the average prediction (baseline) should equal the actual model prediction for that instance.

(b) **Missingness**
If a feature is missing in the model (i.e., it has no impact on the prediction), then the SHAP value for that feature should be zero.

(c) **Consistency**
If a model changes in a way that increases the marginal contribution of a feature (holding all else constant), the SHAP value for that feature should not decrease.

These three principles ensure that SHAP results are faithful. The various approximations provide us with different algorithms approximating the Shapley values.

(a) **KernelSHAP**
KernalSHAP is the first method [34] that estimates Shapley values by solving a weighted linear regression problem. This method involves sampling instances from the feature space and using a Lasso regression model to estimate the Shapley values. The Lasso model is a linear model that includes an L1 penalty term, which helps with feature selection and provides a sparse explanation.

The KernelSHAP algorithm involves the following steps:

i. Generate a dataset of binary-masked instances by randomly selecting combinations of features.

ii. Calculate the output of the black-box model for each masked instance.

iii. Fit a weighted linear regression model on the generated dataset, where the weights are determined by the similarity between the masked instance and the instance of interest.

The coefficients of the linear regression model obtained represent the approximate Shapley values for each feature.

(b) **TreeSHAP**
In contrast to KernelSHAP, TreeSHAP also introduce by [34] calculates the exact Shapley values for each feature by recursively traversing the decision tree, assigning contributions to each feature as it descends the tree. It uses a dynamic programming approach to avoid redundant calculations and reduce

computational complexity. However, this method is specific to tree-based models.

The TreeSHAP algorithm involves the following steps:

i. Traverse the tree from the root to the leaf nodes, recording the decision path for the instance of interest.

ii. Assign contributions to each feature encountered along the path, taking into account the number of possible feature combinations and the probability of each combination.

iii. Repeat the process for all trees in the ensemble, if applicable.

iv. Average the contributions across all trees to obtain the final Shapley values.

Some other optimizations have been proposed by [34], such as DeepSHAP, LinearSHAP, Sampling SHAP, and Partition SHAP. However, since these methods are linked to specific models that are either not sufficiently performant or not suitable for the project (e.g., models used for image classification or text recognition), these methods seem less useful in our context.

Other interpretability methods exist; however, they are less recognized in the field. For example, Individual Conditional Expectation (ICE) often provides results that are not always reliable, which can be problematic in areas where each decision can result in significant costs, such as fraud detection. Other methods like Counterfactual Explanations provide multiple different explanations for a single model, which can require a lot of time to analyze in order to make a correct decision, making it difficult to implement in real-world scenarios.[6]

2. **LIME (Local Interpretable Model-agnostic Explanations)**

LIME [35] is based on the idea that even if a global model is complex and difficult to interpret, it is possible to approximate it locally with a simpler and interpretable model. The main steps of LIME are as follows:

(a) To explain a specific prediction, LIME generates a set of synthetic data around the instance of interest by perturbing its features.

(b) The model is used to predict the outcomes for the generated synthetic data.

(c) Each synthetic instance is weighted based on its similarity to the instance of interest (using distances such as Euclidean distance or cosine similarity).

(d) A simple and interpretable model (such as linear regression or a decision tree) is fitted on the weighted synthetic data.

(e) The coefficients of the interpretable model are used to explain the prediction of the instance of interest, indicating the importance of the local features.

However, LIME has several drawbacks that make it less preferable compared to SHAP for local explanations of black-box models:

(a) LIME assumes that models are locally linear, which is not always the case.

[6]Interpretable Machine Learning, Christoph Molnar, 2024-05-26, last check 11/07/2024 on https://christophm.github.io/interpretable-ml-book

(b) Unlike SHAP, LIME, being a newer approach, is not yet optimized and therefore takes a long time to compute.

(c) LIME is not stable [36], meaning that for the same prediction, some results may differ from one instance to another.

(d) LIME is sensitive to adversarial attacks [37] , which could disrupt the interpretation of frauds and complicate legal proceedings.

Due to these drawbacks of LIME, we prefer to use SHAP for black-box model explanations in this project.

3 Methodology

In this chapter, we will present the approach used for fraud detection. The previous chapter served as a starting point on the methods addressed in this research. We will provide a step-by-step explanation of our method, including the details of each step and the underlying concepts that inform our approach.

This method consists of main steps: data preprocessing, imbalance management, feature engineering, model selection and optimization, model explainability and interpretability. We will describe each of these steps in detail below.

3.1 Preprocessing

As mentioned in Section 1.5, we will use the dataset provided by the Université Libre de Bruxelles in this research. This dataset, having only been anonymized, is still raw and preprocessing needs to be performed.

First, we conducted an analysis of the raw data. This allowed us to notice that there are no missing values. However, among the data, we found 1081 duplicate entries, including 19 in the fraudulent class. These duplicates were removed to avoid any bias that could be introduced into the models. In fact, if some duplicate transactions are in both the training and testing sets, the evaluation of the model will be biased and it could be prone to overfitting.

Next, we needed to divide the dataset coherently. Since the data is time-related, we chose to split the dataset based on time, allowing for a more logical training and use case context. Additionally, we needed to reserve a portion of the data to generate a continuous data flow for the application that will serve as a POC (proof of concept).

We removed the last 20% to become our dataset use to simulate the data flow. From the remaining 80%, we kept 20% for the test set and 80% for the training set. All validation will be made with a Kfold cross validation.

Outlier management was considered. We tried two different approaches: a Z-score approach and an Isolation Forest approach. However, certain frauds are often considered outliers. It is therefore important to consider whether it is appropriate to remove these outliers. An evaluative study will be conducted in Chapter 4. In fact, some methods even rely on this concept for fraud detection [38].

We then performed several data scaling techniques, studying both standard scaling, min max scaling and robust scaling. In our case, robust scaling seems to be the preferred choice. Since the outliers are retained, it is better to use this scaler to prevent the data from being compressed during scaling, which might occur with standard scaling or min max scaling. However, the results of our models did not differ significantly between using standard scaling, min max scaling or robust scaling, so we will consider both cases simultaneously, preferring robust scaling when results are identical due to its more appropriate properties in this context.

Finally, we studied the possibility of multicollinearity between our independent variables. As shown in Figure 18, the variables appear to be independent, indicating no multicollinearity issues (consistent with the dataset's description of being anonymized by PCA). This is confirmed by the fact that none of the variables have a variance inflation factor

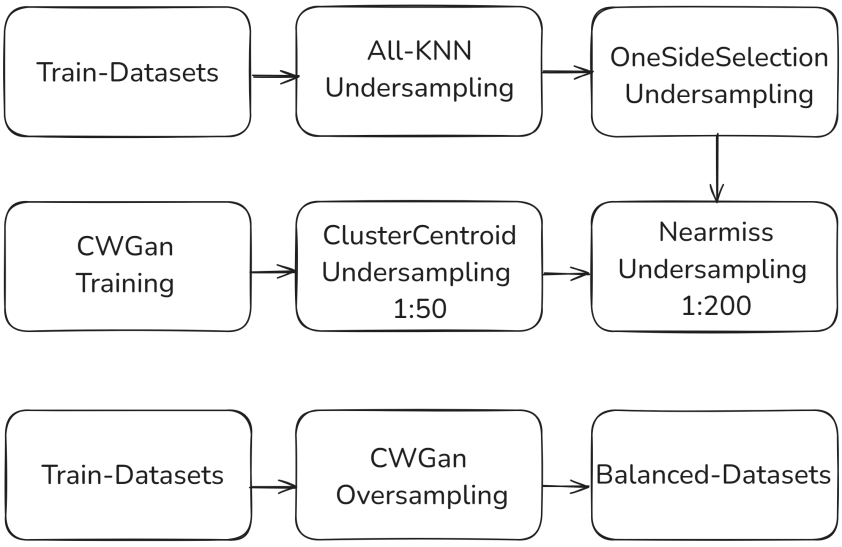


Figure 5: Sampling steps for training GANs

score greater than 5, except for Amount, as shown in Table 8. However, since Amount is one of the two non-anonymized and typically important variables in prediction, we decided to retain it for now but will pay particular attention to it during feature selection.

3.2 Imbalanced Management

As explained in Section 1.5, the dataset is highly imbalanced. This imbalance directly causes performance issues in the models. We noticed that certain methods worked better with some models and not with others. Here, I will detail the different preprocessing techniques used for imbalanced management in this research to develop performant models. To compare the different methods, we will use our three most recognized and performant machine learning models, namely Light Gradient Boosting, Extreme Gradient Boosting, and Random Forest.

3.2.1 Hybrid Sampling

The first method we employed, which provided consistent results with models such as CATBoost and XGBoost, is a hybrid approach involving both under-sampling and an over-sampling method called cWGAN [12], which was explained earlier in Section 2.2. Noted that the data was preprocessed and scaled before the following steps. The proposed solution's schema can be found in Figure 5.

In our research, we noticed that training the cWGAN directly on the dataset was extremely computationally expensive and less performant than simply not performing any sampling. Due to this, we performed an undersampling phase before training the cWGAN. Initially, we found that undersampling methods such as OSS and A11KNN worked best with algorithms like XGBoost and CatBoost [27]. In addition to these two methods, we tested ClusterCentroids, being one of the most recognized methods, as well as NearMiss-Sampling, which reduces training time without diminishing model performance.

Information on the distribution of new sampled values, GAN training, and loss, along with various graphs on optimizing the sampling ratio, can be found in the Figure 28 and Figure

29. We can notice that some distribution approximated by the c-WGAN like the one for the variable V12 seems not to fit perfectly the distribution. Besides, different trials by modifying the hyperparameters we always had one or two variables that is not approximate perfectly. However, these distorted distributions don't always seem to negatively influence our results with the data sampled by this method. It may still be worth investigating another type of GAN that could better approximate our distributions. It's also worth noting that this c-WGAN chose the wassertain distance to determine the error made by the generator. Indeed, previous studies have shown that a distance achieving a smaller error tends to decrease the variance of the error committed by the generator, which complicates its training and therefore its overall performance. For more information, please read Justin Engelmann's paper [12]. Once the cWGAN was trained, we could generate data at will. We studied different sampling ratio possibilities with our model. By default, the ratio of minority to majority classes after resampling is 1:1, meaning we perform oversampling until we have as many fraud cases as non-fraud cases. However, this proportion may not be optimal. From our research on the optimum sampling ratio, we found that adjusting this ratio could slightly improve predictions (by one or two correct predictions compared to the default ratio), but the optimal ratio depended directly on the initial sampling. Since GAN sampling is random, the optimal sampling ratio is also random. Due to all these deductions, we decided to retain the default sampling ratio of 1:1.

Additionally, since the cWGAN involves a learning step based on the loss of a machine learning model, we tested our most performant models, such as LightGBM and XGBoost, in addition to RandomForest, which is the base model. However, no significant differences were observed.

3.2.2 Cost-sensitive Approach

As discussed in Section 2.2, the second method to manage imbalanced data is the cost-sensitive approach. Among the various approaches, the relabeling method did not provide satisfactory results with our models. Conversely, Instance Weighting and Direct Approach yielded very good results with LightGBM as we can see in Table 2.In the case of the Direct Approach, we performed hyperparameter tuning with GridSearch to find the ideal weights to assign to the loss function of our model. This tuning was conducted simultaneously with the tuning of the model's hyperparameters. In this context, the ideal weight is $\{0:1, 1:8\}$.

Model	Accuracy	F1	Precision	Recall	ROC AUC	PR AUC
LightGBM	0.999736	0.915493	0.984848	0.855263	0.971832	0.842547

Table 2: Cost-sensitive direct Aproach for Light GBM

We also considered the Instance Weighting technique and found that the optimal choice was to assign a weight of 1 to non-fraud cases and a value of 8 to fraud cases, which is equivalent to the Direct Approach. When we attempt to modify the weights for instances where the model makes errors, we observe that while sometimes the model successfully detects these frauds, it introduces errors in numerous other transactions. It might be interesting to retain the concept of this approach when we aim to clearly identify a specific type of fraud, allowing us to assign more weight to these fraud cases.

3.3 Feature Engineering

In this section, we will consider the various manipulations that can be performed on features.

3.3.1 Creation of New Features

In the context of machine learning, creating new features from existing ones can be a wise choice to add information or simplify the model's learning process by, for example, reducing dimensions. Several techniques are known to apply this idea.

1. **Feature Interaction** It is possible to highlight feature interactions through various statistical methods such as mean, variance, entropy, etc. Unfortunately, in our case, not knowing the actual values of the data due to anonymization via PCA, it is difficult to choose which variables to combine.
2. **Polynomial Features** Another possibility is to perform polynomial regression to extract information from multiple correlated features. Here, as we have seen, our features are hardly correlated, so this method does not seem feasible.
3. **Dimensionality Reduction** Since the variables are already principal components from PCA, performing dimensionality reduction on already reduced dimensions does not seem useful. This idea was proven during some of our inconclusive tests.
4. **Binarization** Another idea is to provide information on a threshold for variables. Unfortunately, since the variables are anonymized, it is difficult to use thresholds when we do not know the physical values behind them.

3.3.2 Feature Selection

As seen in the paper "machine learning based credit card fraud detection using the GA algorithm for feature selection" [24], selecting a subset of features to extract the most useful information from this model can provide better performance in terms of model accuracy or computation time.

Different methods were considered in this project to select the best features.

1. **Chi-Square** We performed a feature selection of the top k features based on a chi-square score.
2. **Select From Model** We selected features based on the weights associated with the prediction of variables in a machine learning model. We chose the Random Forest model, being one of our very good models but not the most performant, to avoid overfitting on our best models.
3. **Info-Gain** We performed a feature selection of the top k features with the highest information gain score.

For each of these methods, we decided to retain the top 3 subset of features for our models.

The results of these different feature selections can be found in Table 3

3.4 Model Selection

This section is dedicated to exploring the models utilized in this project. These investigations were initially guided by my knowledge and subsequently refined as my research in the field progressed. Interpretable models will be discussed in the next section.

During my research, I learned that ensemble models are the most appropriate for classification tasks on imbalanced datasets. Therefore, ensemble models were prioritized in this study.

3.4.1 Bagging

As mentioned, the first ensemble models I used were based on my knowledge from courses such as Machine Learning [1] and Machine Learning in Space Science [39]. At first, I explored models using the Bagging principle, which is based on three main steps:

1. Create multiple subsets of data by sampling with replacement from the original dataset.
2. Train a distinct model on each subset.
3. Combine the predictions of all models by averaging (for regression problems) or majority voting (for classification problems).

The model that clearly uses this concept is Random Forest, where each distinct model is a decision tree. Another model adapting the Random Forest technique is Extra Trees. These models do not perform bootstrapping on the dataset but use the entire dataset, which reduces the bias introduced by this method. The Extra Trees model then splits nodes randomly, which reduces variance compared to Random Forest, which splits nodes to choose the best division.

3.4.2 Boosting

The second method explored is Boosting. This method, already covered in machine learning courses and in the context of fraud detection, as discussed in the Related Works section, is based on the idea of sequentially placing simple models, where each model attempts to correct the errors of the previous models. These models, placed sequentially, give more importance to errors made by previous models so that the final decision is a weighted sum of the different models.

Among the models using this Boosting phenomenon, the most performant are Extreme Gradient Boosting (XGB) [40], Light Gradient Boosting (LGB) [41], and Categorical Boosting (CatBoost) [42]. These three models incorporate different approaches to the Boosting concept. Extreme Gradient Boosting is the first model to significantly improve Gradient Boosting models and is by far the most popular model on the market.

LightGradientBoosting, developed by Microsoft, aims to optimize training time with two methods:

1. **Gradient-Based One-Side Sampling (GOSS):** The idea of GOSS is that data instances with higher gradients contribute more to information gain. To maintain the accuracy of the information, GOSS retains instances with higher gradients and performs random sampling on instances with lower gradients.

2. **Exclusive Feature Bundling (EFB):** To reduce dimensionality, improve efficiency, and maintain accuracy, EFB bundles these features into an "Exclusive Feature Bundle".

Categorical Boosting also brings modifications to the Boosting principles. The main improvements are:

1. **Symmetric Trees:** CatBoost uses level-wise growth for trees, however these trees are symmetric, unlike XGB. This balanced tree architecture aids in efficient CPU implementation, thus reducing training and prediction time.
2. **Ordered Boosting:** CatBoost counters prediction shift, which is defined as overfitting when the dataset is too small.
3. **Native Feature Support:** CatBoost automatically handles the conversion of data regardless of whether it is numerical, categorical, or even text.

Although XGBoost currently outperforms LightGBM, which in turn outperforms CatBoost, the comparison between these models should be done in each case. The performance of these three models depends directly on the datasets used and will be studied simultaneously in this project. However, since CatBoost and LightGBM are faster to train and predict, we may prefer one of these models in production.

3.4.3 Stacking

The last method for creating ensemble models is called Stacking. This method collects the predictions of different models and uses this information as input to generate a final prediction. Being more flexible, this method requires a detailed study to determine the specific models to choose in the first stage of stacking.

3.5 Interpretable Model

As explained in the Related Works, the pursuit of interpretability is essential in domains where predictions can be critical, such as fraud detection.

To evaluate the performance of our more complex interpretable models, we used the Logistic Regression model as a reference, being a simple and interpretable model. This model allows us to compare whether our interpretable models significantly reduce accuracy in the quest for interpretability.

Subsequently, we utilized another simple and interpretable model, which is decision trees. Given that ensemble methods using decision trees provide good results, it is interesting to use them here.

Finally, we employed the Explainable Boosting model [43], which utilizes research conducted on GA2M [32]. This method allows the model to be interpretable while only observing a 0.01 decrease in the F1 score. Recall that GA2M seeks to explain the model by:

$$g(E[y]) = \beta_0 + \sum f_i(x_i) + \sum f_{i,j}(x_i, x_j)$$

To achieve this, the model follows several key steps represented in Figure 6.

1. First, the model uses a boosting procedure where each step of the training recursion is limited to access only one or two variables at most, in an alternating manner.

It aims to learn the best function f_j for each variable. This step corresponds to a line in Figure 6. During this learning process, the learning rate is kept very low to ensure that the order in which variables are addressed does not introduce bias.

2. Next, the model employs the bagging method. The model performs the boosting procedure limited to each variable a large number of times. Finally, it determines f_j by averaging the results (values in the case of regression and probabilities of belonging to a class in the case of classification) to obtain graphs that allow for great interpretability of our results. This step corresponds to the greens boxes in Figure 6.

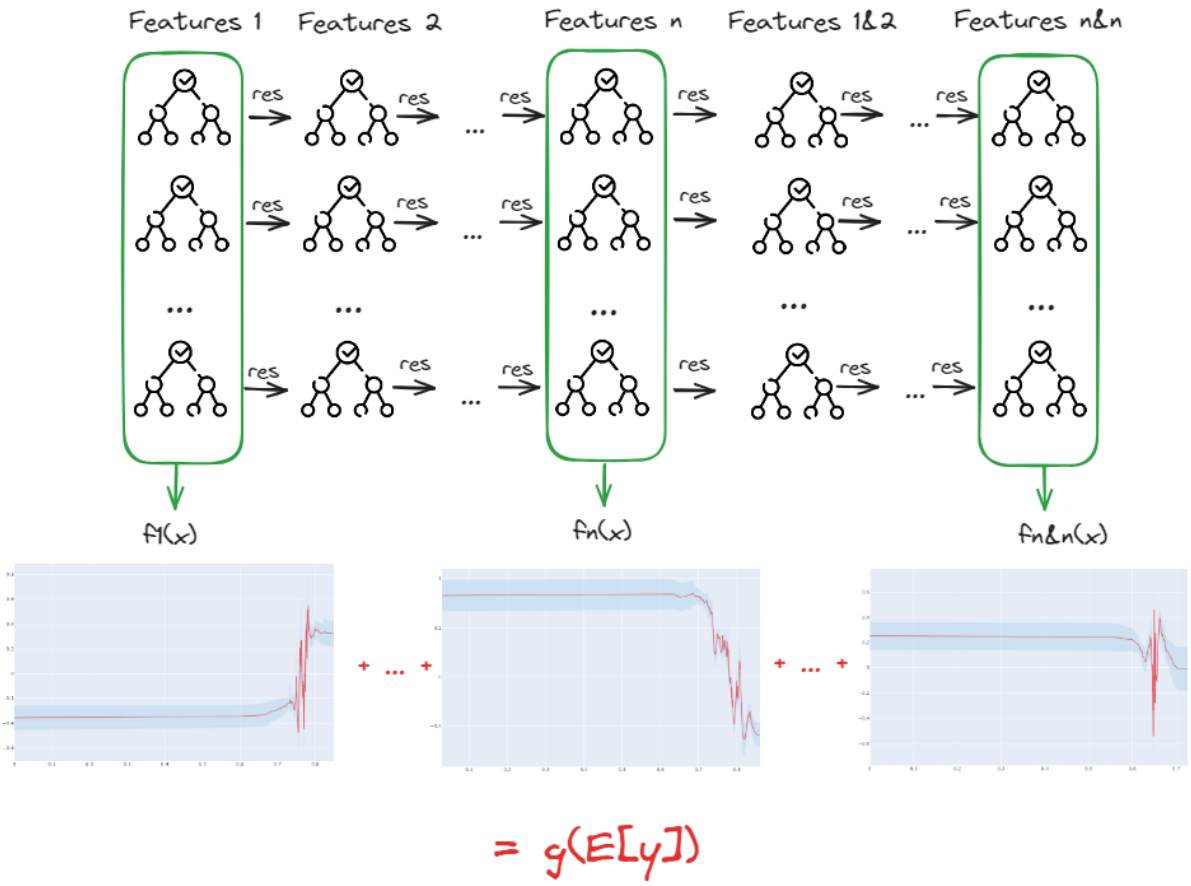


Figure 6: Main steps of Explainable Boosting [43]

Having access to the final graph f_j, representing how the model actually predicts, allows us to obtain both global and local interpretations.

The various graphs associated with the interpretation of different variables can be found in the appendix.

3.6 Explainable Method

For the models mentioned above in Section 3.4, we applied the SHAP library to our trained model to obtain explanations of the predictions. For more information on SHAP, we invite you to read the Related Works Section 2.4.4. The Evaluations of the differents plot can be found on Section 5.1

4 Evaluation

In this chapter, we will begin by introducing the different metrics used for model evaluation. We will then discuss these metrics in relation to the topic of our report, which focuses on Fraud Detection with an imbalanced dataset and varying error costs associated with prediction errors. Following this, we will evaluate different preprocessing methods using the three best-performing models we have identified—Extreme Gradient Boosting, Light Gradient Boosting, and Random Forest—along with our most interpretable model, Explainable Boosting.

Next, we will assess the models discussed in the previous chapter (Section 3.4) using both sampling methods and cost-sensitive learning approaches. In the final two sections of the evaluation part (sampling and cost-sensitive methods), the models have been fine-tuned using GridSearch with a cross-validation method where k is equal to 5. The different hyperparameter values tested can be found in the notebooks related to this thesis. The most significant results will be presented in this report, either directly in the text or in the appendix. However, numerous tests with lesser performance will not be included, but they are available in the notebook "final-modelling."

4.1 Metrics

In this part of the project we will define the different metrics we will use.

4.1.1 Accuracy

Accuracy is the most straightforward metric, representing the ratio of correctly predicted instances to the total instances in the dataset.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Population}}$$

While accuracy is useful, it can be misleading for imbalanced datasets where the number of instances in each class is not equal.

4.1.2 AUC (Area Under the Curve)

AUC refers to the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is a plot of the true positive rate (sensitivity) against the false positive rate (1-specificity). This value represents the likelihood that a randomly chosen positive instance is ranked higher than a randomly chosen negative one. A model with a ROC AUC of 0.5 is no better than random guessing, whereas a model with a ROC AUC of 1.0 indicates perfect classification. However, similar to Accuracy, this score can be misleading when dealing with imbalanced datasets since it could yield good results even if all the frauds are misclassified.

4.1.3 Precision

Precision, also known as positive predictive value, measures the ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

High precision indicates a low false positive rate, which is particularly important in cases where false positives are costly (e.g., spam detection).

4.1.4 Recall

Recall, also known as sensitivity or true positive rate, measures the ratio of correctly predicted positive observations to all observations in the actual class.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

High recall indicates that most actual positives are identified by the model, which is crucial in scenarios where missing positive cases is costly (e.g., disease detection).

4.1.5 F1 Score

The F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both concerns.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is especially useful when the class distribution is imbalanced, as it considers both false positives and false negatives.

4.1.6 PR AUC (Precision-Recall Area Under Curve)

The PR AUC measures the area under the Precision-Recall curve, which plots precision against recall for different threshold values. This metric is particularly useful for imbalanced datasets[7]. A higher value indicates better performance, with a value of 1.0 representing a perfect model. Unlike ROC AUC, PR AUC focuses on the performance of the positive class.

These metrics together provide a comprehensive picture of a model's performance, helping to make informed decisions on model selection and improvement.

4.2 Cost Adapted Metrics

As we have seen, Accuracy and ROC AUC can lead to misleading results in our project due to the imbalanced dataset. However, since these metrics are widely used in research, we will include them for comparison purposes, but we must remain critical of their values. Let's now take into account the cost behind the prediction of our model.

[7]Ultimate Guide to PR-AUC: Calculations, uses, and limitations Tom alon, April 8, 2024, Last checked 11/07/2024 on https://www.aporia.com/learn/ultimate-guide-to-precision-recall-auc-understanding-calculating-using-pr-auc-in-ml/

The important aspect of this problem is that the costs associated with different classification errors are not the same. Indeed, not detecting bank fraud is often much less serious than wrongly accusing someone. Indeed, while letting a fraud go through will "only" cost the value of the transaction, wrongly accusing someone can incur enormous costs, including the potential loss of the client, loss of trust among other clients aware of the bank's mistake, and legal fees from a lawsuit in which the bank is likely to lose. On average, the costs incurred by a bank transfer fraud amount to €4782 in France in 2022[8] compared to several tens of thousands of euros for wrongly accusing someone in justice according to Intech. If we look at the cost function of our fraud detection problem[9]:

$$J(y, \hat{y}) = C_{fp} \times (1 - \hat{y}) \times y + C_{fn} \times \hat{y} \times (1 - y)$$

where:

1. y is the true binary variable (1 for fraudulent, 0 for legitimate)
2. $\hat{y}$ is the predicted binary variable (1 for fraudulent, 0 for legitimate)
3. C_{fp} is the cost of a false positive (i.e., the cost of wrongly accusing a non-fraudulent transfer of being fraudulent)
4. C_{fn} is the cost of a false negative (i.e., the cost of not detecting a fraudulent transfer)

In this equation we admit that the Cost of a good prediction is equal to 0

We can rewrite this cost function in terms of False Positives and False Negatives as:

$$J(y, \hat{y}) = C_{fp} \times FP + C_{fn} \times FN$$

However, since we know that C_{fp} is much higher than C_{fn}, and in our case, we aim to achieve the least amount of error possible. Our use case suggests that the costs are generally much higher than the number of errors made by our model. For all these reasons, we can approximate by saying that C_{fn} is negligible.

$$J(y, \hat{y}) \sim C_{fp} \times FP$$

Minimizing a constant (here we suppose that the cost is the average cost and so is constant over a period of time where the model is trained) times a variable is equivalent to minimizing the variable. Moreover, since $TP + FP = P$, which is constant, minimizing FP is equivalent to maximizing TP. Let's get the weighted value maximization of TP over all positive values.

$$P = \frac{TP}{TP + FP}$$

[8] Les entreprises face à la menace de la fraude au virement bancaire, 19/12/2023, Last checked 11/07/2024 on https://rendre-notre-monde-plus-sur.goron.fr/les-entreprises-face-a-la-menace-de-la-fraude-au-virement-bancaire/

[9] Fraud detection with cost-sensitive machine learning by Roman Moser in Towards Data Science, 29/03/2019. Last checked: 10/07/2024 https://towardsdatascience.com/fraud-detection-with-cost-sensitive-machine-learning-24b8760d35d9

where we redefine precision.

This shows clearly that if we want our model to achieve the best results in the scope of fraud detection, we should optimize the precision.

In the previous demonstration, we approximated that the cost of false negatives is negligible compared to that of false positives. However, the costs of false positives can vary greatly depending on the specific transfer. For instance, if the transfer is internal (within the same bank), the costs will be much lower if a transfer is wrongly denied for potential fraud. Similarly, depending on the transfer, the costs of false positives can vary significantly. For example, there could be a loss of customer trust, or if an individual sees that a type of fraud is not being blocked, they might attempt that kind of fraud again.

To address this, let's define θ as the ratio between the costs of false negatives and false positives. At equilibrium, we aim to find θ such that

$$\theta = \frac{C_{fn}}{C_{fp}} = \frac{FN}{FP}$$

while keeping it as small as possible.

Therefore, if we seek to minimize $C_{fp} \times FP + C_{fn} \times FN$ while maximizing the number of correct predictions, we need to aim to maximize the F1 Score. Indeed, starting from the formula explained in Section 4.1.5, we have

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

By substituting *Precision* and *Recall* with their values as expressed in Section 4.1.3 and Section 4.1.4:

$$F_1 = \frac{2 \times \frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

This gives us

$$F_1 = \frac{2 \times \frac{TP^2}{(TP+FP)\times(TP+FN)}}{\frac{TP\times(2TP+FP+FN)}{(TP+FP)\times(TP+FN)}}$$

Which simplifies to

$$F_1 = \frac{2 \times TP}{2 \times TP + FN + FP}$$

This proves that this metric is optimal when TP is maximized and $FN+FP$ is minimized. However, here we seek to minimize $C_{fn} \times FN + C_{fp} \times FP$, where $\frac{C_{fn}}{C_{fp}} \times FN + FP$.

The F1 score is a particular case of the F-beta score, which is defined as:

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{\beta^2 \times P + R}$$

which can be rewritten as:

$$F_\beta = \frac{TP}{TP + \frac{1}{1+\beta^2}(\beta^2 \times FN + FP)}$$

This metric will be optimal when $\frac{1}{1+\beta^2}(\beta^2 \times FN + FP)$ is minimal. Therefore, if $\beta^2 = \theta = \frac{C_{fn}}{C_{fp}}$, the metric will indeed seek to minimize our weighted cost function for frauds $\frac{C_{fn} \times FN + C_{fp} \times FP}{C_{fn} + C_{fp}}$.

In conclusion, the idea will be to optimize the Precision or the F1 Score if we do not have the different values of the cost (in function if we want to make the approximation or not), or to optimize the F_β Score if the costs are known.

4.3 Preprocessing

In this section, we will evaluate the different preprocessing methods we developed in Section 3.1. To do this, we will evaluate four reference models selected based on their results and the methods they employ. We selected two boosting methods (XGBoost, which seems to perform well with sampling methods, and Light Gradient Boosting, which is effective with cost-sensitive methods), Random Forest as a bagging method, and Explainable Boosting as an interpretable model. We can look at the results on the dataset without any preprocessing in Table 40.

As explained in Chapter 3, we implemented outliers management. However, as seen in Tables 11 and 12, removing these outliers significantly impacts the results of our reference models. Therefore, the following results will include these outliers.

Next, we examined the different scalers available. Although logarithmic scaling in the presence of outliers might remove some information about the extreme values that some frauds might have, thus negatively impacting our models. As mentioned in Section 3.1, no differences seem to emerge between the different scalers for Random Forest, Extreme Gradient Boosting, and Explainable Boosting models, as seen in Tables 13, 14, and 15. However, for Light Gradient Boosting, we observe that results vary significantly depending on the scaler used. The results are not very conclusive at the moment, probably due to data imbalance. We will need to look at the differences for Light Gradient Boosting in the final model evaluation.

4.3.1 Feature Selection

As explained in Section 3.3.2, performing feature selection on our dataset can provide better results. A study was conducted to determine if this could provide better or at least similar results to those obtained with all variables, but with optimized computation time. To select the variables to keep, we used SelectKBest, InfoGain, and SelectFromModel functions from the SKlearn library. Among numerous tests, which can be found in the notebooks, it is noted that InfoGain appears to be the method that selects the most effective variables. However, it is also noted that removing even one variable results in a decrease in the F1 score by at least 0.02. After that, we can retain up to 6 variables without diminishing significantly the results. The best results can be found in Table 3.

It should be noted that the best results between each selection of variables in terms of F1 scores are achieved with the subset Vb or a parent of this subset. It can be observed that, once we remove a variable, the best results are at most equivalent to Vb as long as the subset considered is a parent (i.e., containing at least the variables included in Vb). Additionally, the training time primarily varies for the EBM model, from 2 minutes 25

seconds to 50 seconds for Vb. For the other models, training time differences are less noticeable, with only a 1 to 2 seconds difference.

	Subset	Method and Model	F1
Va	V4, V10, V11, V12, V14, V17	Select From Model on RF	0.863
Vb	V3, V4, V9, V10, V11, V12, V14, V16, V17, V18	InfoGain with RF	0.872
Vc	V4, V7, V10, V11, V12, V14, V16, V17	Select From Model on EBM	0.866

Table 3: Table of the three best subsets found in our feature selection studies

4.4 Sampling

As explained in Sections 2.2 and 3.2, several sampling methods were considered. All the undersampling and oversampling approaches that provided correct results are visible in Tables 16 to 39. Each methods can be found on imbalanced-learn library. These evaluations are based on default values for the different models. Among these models, we selected the top 10 results for all models. The results can be found in Table 4. As we can see here, the Random Oversampling method surprisingly performs very well, with the best precision for Random Forest and the best recall for XGBoost. It also provides the best F1 score when we scale our data using a robust scaler. However, it is also worth mentioning that BorderlineSMOTE is the sampling method which provide the best results.

In addition to the libraries provided by imbalanced-learn, we considered the conditional Wasserstein GAN discussed in Section 3.2. We tested this method on numerous models such as Logistic Regression, K-Nearest Neighbors, Decision Tree, CatBoost Classifier, Light Gradient Boosting Classifier, Explainable Boosting Classifier, Extreme Gradient Boosting Classifier, and Random Forest. The different results can be seen in Table 5. In our various tests, we obtained correct results for XGBoost and ExtraTrees. Although the results for LightGBM seem satisfactory, the cost-sensitive method provides better results. It is noted that the sampling of GANs is stochastic, as we randomly sample from the distribution approximated by the generator. The results obtained in Table 5 are the best empirical results obtained in 10 trials (it is noted that in 3 out of the 10 trials, we obtained the results shown in Table 5). Finally, tuning was performed via cross-validation with K=5 before evaluating these different models on the validation set. The various parameter values can be easily found in the Modelling Notebook.

Model	AUC	Accuracy	F1	Precision	Recall	PR AUC
Random Forest (Table 20)	0.954	1.000	0.884	**0.970**	0.813	0.891
XGBoost (Table 20)	0.984	1.000	0.872	0.895	**0.850**	0.873
LightGBM (Table 20)	0.974	1.000	0.873	0.885	0.863	0.874
EBM (Table 20)	0.983	1.000	0.880	0.943	0.825	0.884
Random Forest (Table 23)	0.942	1.000	0.877	0.970	0.800	0.885
XGBoost (Table 23)	0.985	1.000	0.882	0.931	0.838	0.884
LightGBM (Table 23)	0.972	1.000	0.868	0.917	0.825	0.871
EBM (Table 23)	0.983	1.000	0.872	0.942	0.813	0.877
Random Forest (Table 27)	0.967	1.000	0.869	0.969	0.788	0.879
XGBoost (Table 27)	0.982	1.000	**0.895**	0.944	**0.850**	**0.897**

Table 4: Top 10 models based on F1 score and precision for sampling methods from imbalance learn

Model	AUC	Accuracy	F1	Precision	Recall	PR AUC
Logistic Regression	0.84	**0.99**	0.79	0.93	0.68	0.81
KNN	0.69	0.99	0.54	0.98	0.38	0.68
Decision Tree	0.82	0.99	0.77	0.99	0.63	0.81
CatBoost	0.83	0.99	0.80	0.99	0.67	0.83
Extratrees	0.92	0.99	0.90	0.94	0.86	0.90
Xgboost	0.93	0.99	**0.92**	**0.99**	0.86	**0.92**
Randomforest	0.91	0.99	0.87	0.98	0.78	0.89
LightGBM	**0.93**	0.99	0.91	0.98	0.84	0.91

Table 5: Performance Metrics of Different Models with the WCGAN oversampling

4.5 Cost-sensitive Learning

As discussed in Section 3.2.2, we implemented a cost-sensitive approach across our different models. During our studies, we observed that the weight assigned in the cost function or distributed to instances via the instance weighting method had to be tuned based on the dataset and the model. After tuning the various models, we found that Light Gradient Boosting seems to benefit the most from the cost-sensitive approach.

The results of simple cost-sensitive learning can be found in Table 6. It should be noted that in this table, the ideal weight for each model was tuned to achieve the best results. Additionally, no hyperparameter tuning is currently provided for Explainable Boosting to perform cost-sensitive learning.

Model	Ratio	AUC	Accuracy	F1	Precision	Recall	PR AUC
LightGBM	1:8	**0.93**	**1.00**	**0.92**	0.97	**0.87**	**0.92**
RandomForest	1:12	0.91	1.00	0.88	0.95	0.82	0.88
CatBoost	1:1	0.92	1.00	0.91	**0.98**	0.84	0.91
XGBoost	1:8	0.93	1.00	0.89	0.93	0.86	0.89
Extra Tree	1:8	0.90	1.00	0.87	0.95	0.80	0.88

Table 6: Performance Metrics of Different Models with Cost Sensitive Ratios

Afterward, a hybrid approach was tested where we first performed sampling at different scales (1:10, 1:20, 1:100) using GAN and ADASYN sampling methods. We then trained a cost-sensitive model by re-tuning the model. It turned out that during tuning, we often obtained the best results with a weight of 1:1, indicating that no modification to the cost function was needed to achieve the best model. Additionally, the results obtained using this method were less effective than simply using GAN oversampling or a cost-sensitive approach.

5 Explanation of the Prediction

In this chapter, we will evaluate the different plots provided by the various interpretable models first, and the plots provided by the SHAP method to explain our so-called Black Box models.

5.1 Interpretable Model

In this section, we will explore the various possibilities for interpretability of our models. As explained in section 3.5, we will address the different explanations provided by Logistic Regression, Decision Tree, and Explainable Boosting models. These three models are available in the interpretML library [43].

5.1.1 Decision Tree

First, we will focus on the decisions that can be explained by a simple Decision Tree. Two functions provided by interpretML allow us to explain this decision tree both locally and globally. The global graph, shown in Figure 7, is quite similar to the local graph in Figure 9. The advantage of the local graph is that we can trace the different decisions made in this tree to classify this transaction. Finally, it is also possible to highlight a variable to see how important it is in the decision-making process. For example, in Figure 8, we can see that variable V14 influences three decisions in this tree. It is also noted that on each of these graphs, the longer the branches, the higher the number of instances classified.

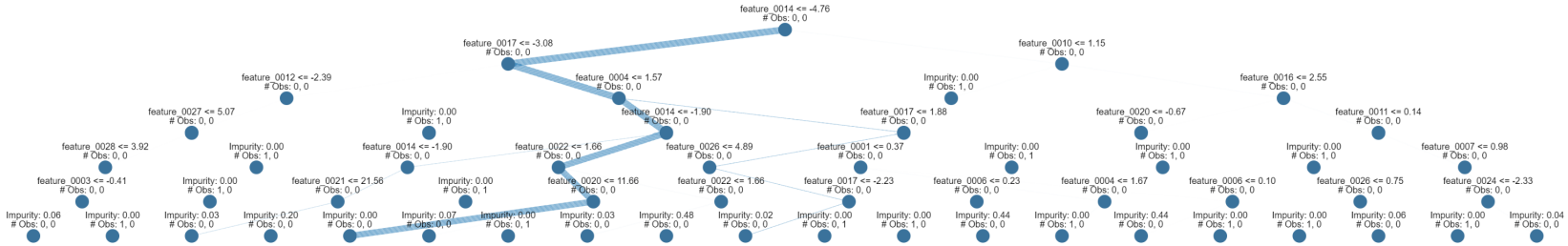


Figure 7: Global explanation

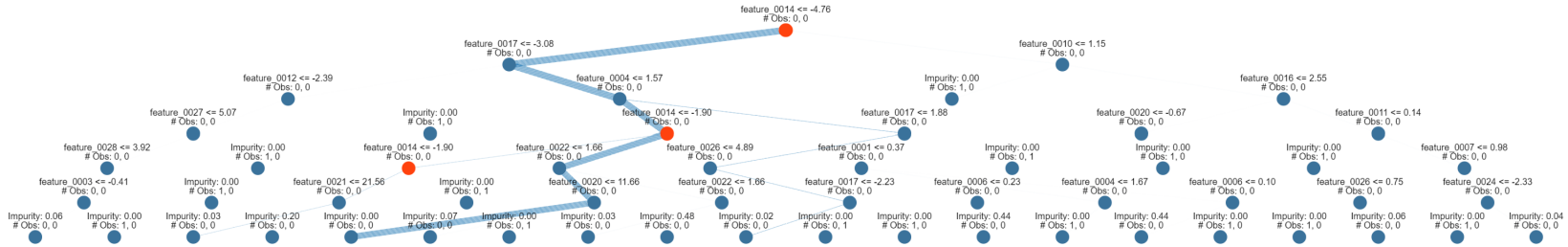


Figure 8: Global explanation highlighting V14

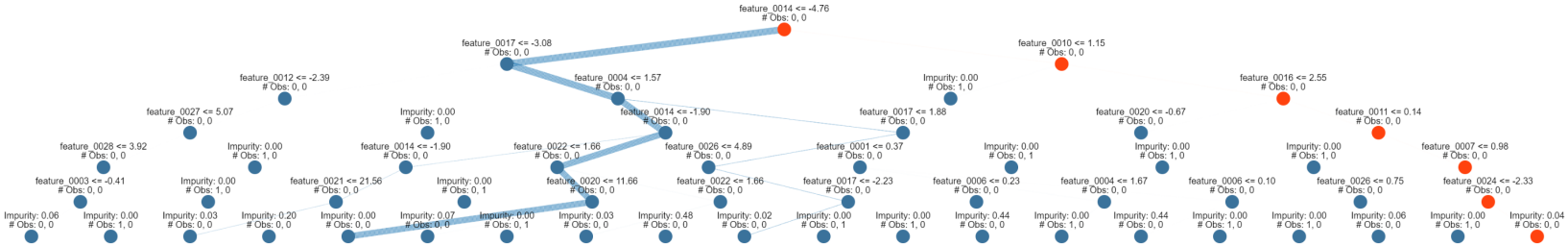
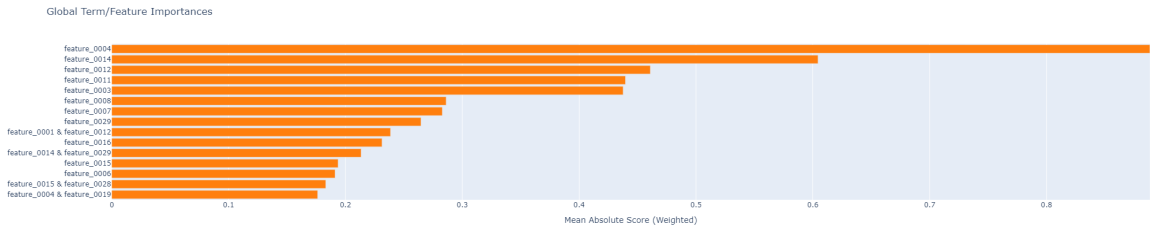


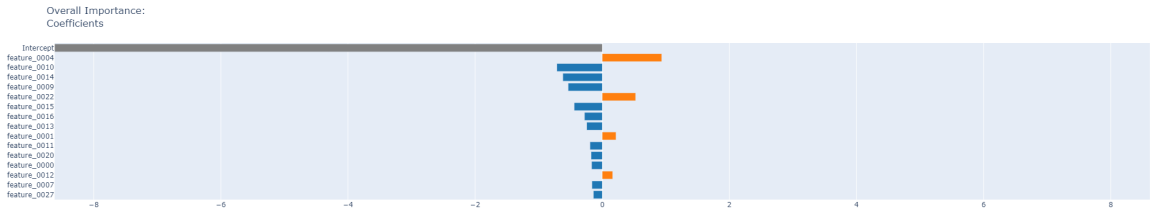
Figure 9: Local explanation

5.1.2 Explainable Boosting and Logistic Regression

Next, we can look at Logistic Regression and compare it directly with Explainable Boosting. As we can see in Figures 10a and 10b, both models provide a global explanation of the results in the form of a Bar Plot. It is noted that as explained in Section 3.5.



(a) Explainable Boosting model



(b) Logistic Regression

Figure 10: Bar plot of Global Explanation

Explainable Boosting also provides the interaction weights of two variables in the predictions. For example, we can see on the Figure 12 the pairwise interaction between v19 and v24. We can see, for example, that if V29 is greater than 1.8 and V4 is less than -2, we have a high probability of being a fraud. Then, the two models allow us to see the evolution of the probability of being a fraudster as a function of the value of a variable, as shown in Figures 11a and 11b. A higher score indicates a greater probability that the model predicts fraud. It's worth noting that the non-linear nature of EBM allows us to reveal what we call critical areas, i.e. areas where we can detect spikes in the probability of falling victim to fraud. For example, in Figure 11c, which is an enlarged version of the graph mentioned above for the variable V1, there is a peak when the value of V1 is between 0.474 and 0.545. It might be useful for an expert familiar with the actual values of these variables to ask why these values are critical, and whether it is possible to identify a specific type of fraud.

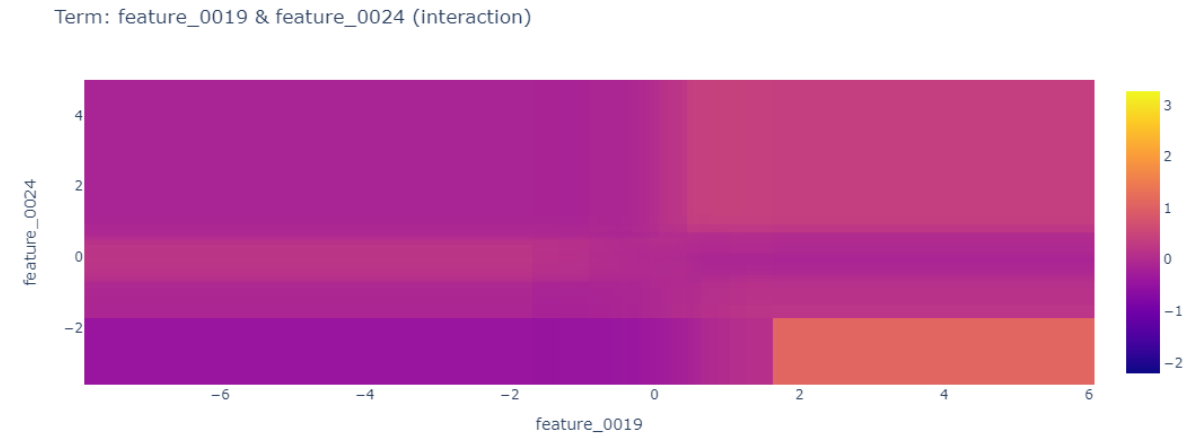
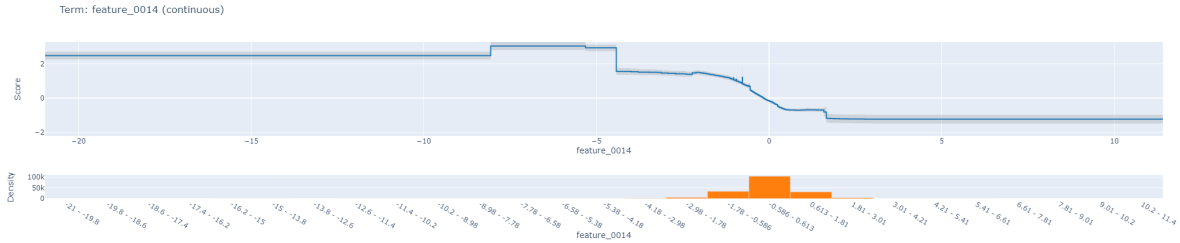
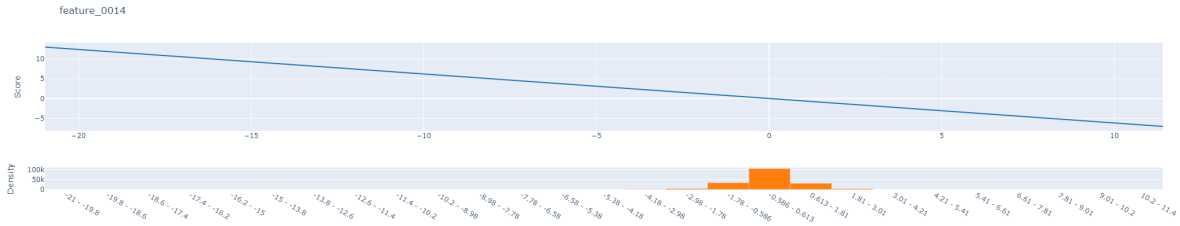


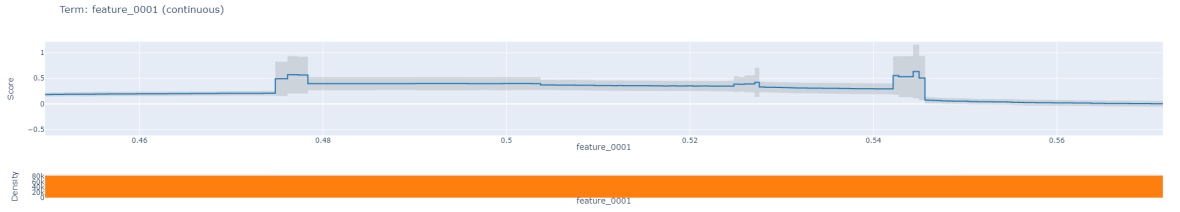
Figure 12: pairwise interaction with FAST on Explainable Boosting



(a) Explainable Boosting model for V14



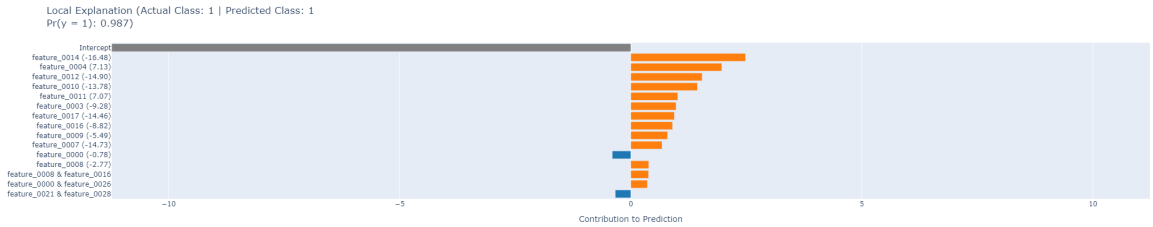
(b) Logistic Regression for V14



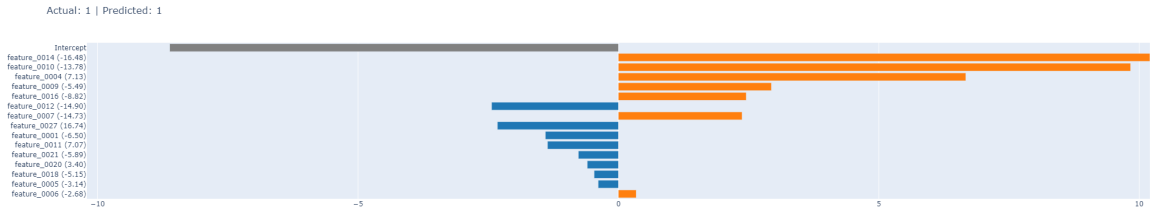
(c) Explainable Boosting model for V1 (zoom)

Figure 11: Global Explanation of a selected variable

Finally, as we can see in Figures 13a and 13b, it is also possible to obtain a local explanation for a given transaction for both Logistic Regression and the Explainable Boosting model.



(a) Explainable Boosting model



(b) Logistic Regression

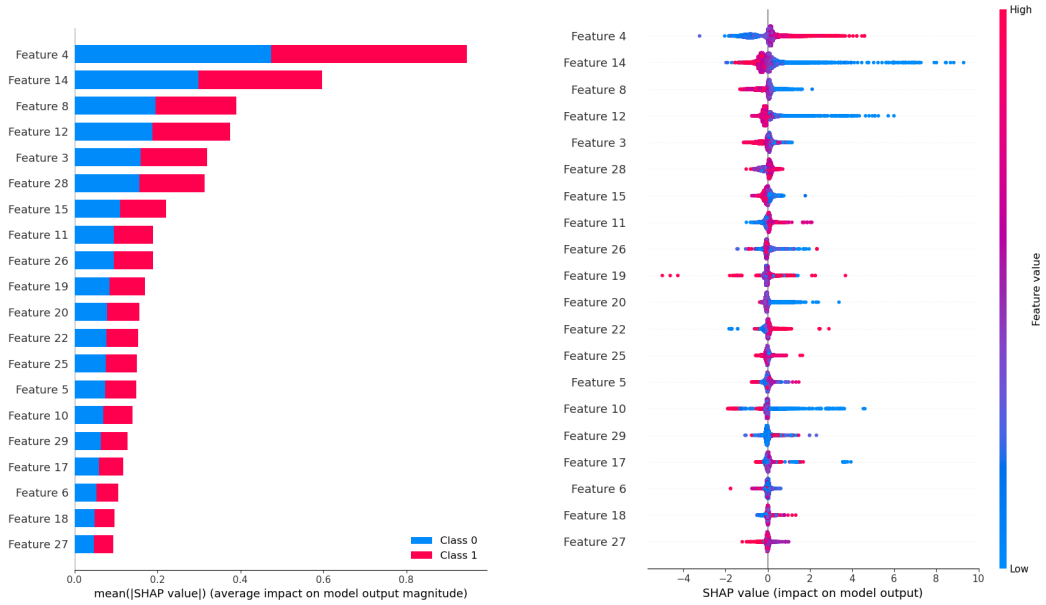
Figure 13: Bar plot of Local Explanation

5.2 Explain Black Box Model with SHAP

The SHAP library provides three main types of plots for obtaining interpretability, local plot, global plot and dependence plot. In this exploration of the SHAP explanation, we will use our best model the Ligth Gradient Boosting model with a Cost sensitive approach. Firstly, SHAP propose two plots for global explanation:

1. **Bar Plots:** The bar plot is probably the simplest form of global explanation. It plots, for each feature, either the mean or the max of the absolute SHAP values for that feature. The blue part represents the weight of this variable in predicting non-fraud, and the red part represents the weight in predicting fraud. We can see an example in Figure 14a, where it is evident that feature 4 seems to influence the model's prediction the most on average.

2. **Beeswarm Plot:** This second plot, shown in Figure 14b, displays all SHAP values for each element in our dataset. Although this plot provides much more information than the bar plot, it can be more complicated to interpret for non-experts. This graph shows the distribution of SHAP values for the data. A negative SHAP value indicates that the variable's value for that instance influences the prediction towards non-fraud, and a positive value influences towards fraud. The colors associated with the points represent the values of the variables. For instance, for feature 14, high values seem to influence the prediction towards fraud. However, the values most influencing the prediction seem to be rather low for this feature, according to the graph. It could be interesting for an expert to explore these variable values to determine if there are critical values for the model (i.e., values where the model automatically predicts fraud). Additionally, it shows that some instances of the variables significantly influence the average value since many SHAP values are close to 0. Finally, we can see that although feature 4 seems to influence the model the most on average, in some cases, feature 14 is the most influential.



(a) Global Bar Plot from SHAP

(b) Beeswarm Plot from SHAP

Figure 14: Global Explanation Plots from SHAP

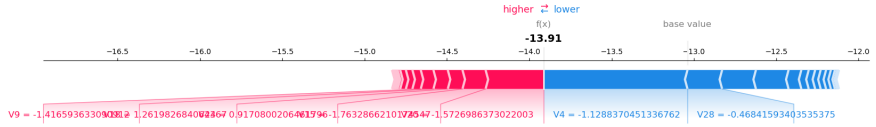
The SHAP library also provides local interpretability for specific predictions. There are 4 functions provided by SHAP:

1. **Force Plot:** The force plot shows the tendency of each variable's value to influence a fraud or a legitimate transfer. Red indicates values influencing fraud, and blue indicates values influencing a legitimate transfer. This plot also shows the variable values for that instance. In the example shown in Figure 15a, we see the explanation for the first prediction on our train set. For instance, the value of variable V4 significantly influences the prediction towards fraud with a value of -1.129.

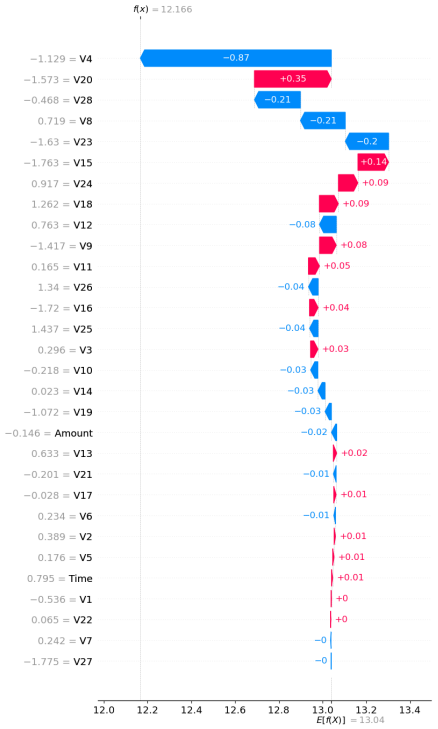
2. **Waterfall Plot:** This plot provides the same information as the force plot but in a more readable format.

3. **Bar Plot:** Again, this plot is another way to highlight the information shown in the waterfall plot or force plot.

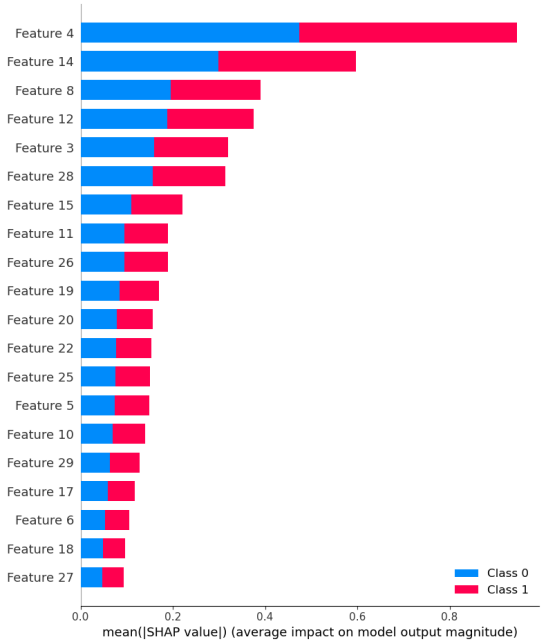
4. **Decision Plot:** This plot shows the cumulative SHAP values for each instance of our variables. The advantage of this plot is that it is possible to plot multiple instances on the same plot, which can be useful for comparing two model decisions based on variable values. For example, in Figure 15d, we see that for two frauds, the variable influencing the model's decision the most seems to be V14 for the first transfer (dashed line on the graph), while the other (solid line on the graph) is mainly influenced by the value of V4.



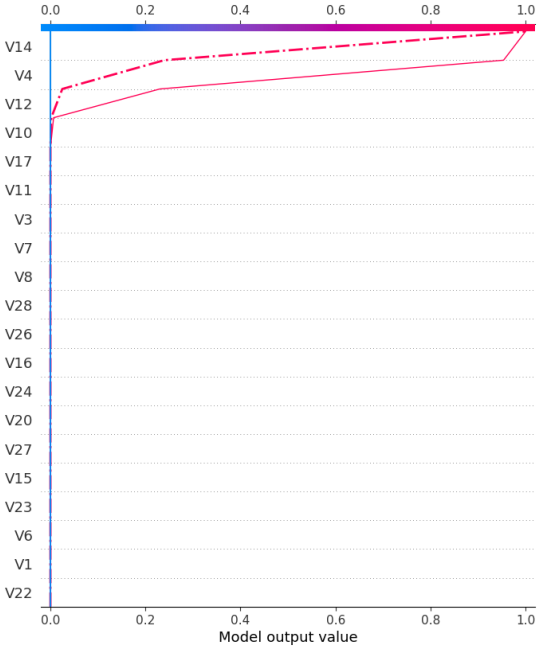
(a) Local Force Plot from SHAP



(b) Local Waterfall Plot from SHAP



(c) Local Bar Plot from SHAP



(d) Decision Plot for 2 Fraud and 2 Legitimate Transfers

Figure 15: Local Explanation Plots from SHAP

Finally, the last type of plot that provides explanations related to the features interaction is the Dependence Plot. This plot provides a global view of SHAP values based on the variable values. Additionally, the color of the points indicates if the feature of interest has interactions that influence the model's decision with other variables. By default, the variable with the most average interactions is chosen, but this can be specified in the function parameters. For example, in Figure 16, we can see that elements with a V3 value around 1 and a high V4 value are likely to be classified as fraud. A more illustrative example can be found with non-anonymized variables in the SHAP documentation.[10] Other dependence plots can be found in Figure 30.

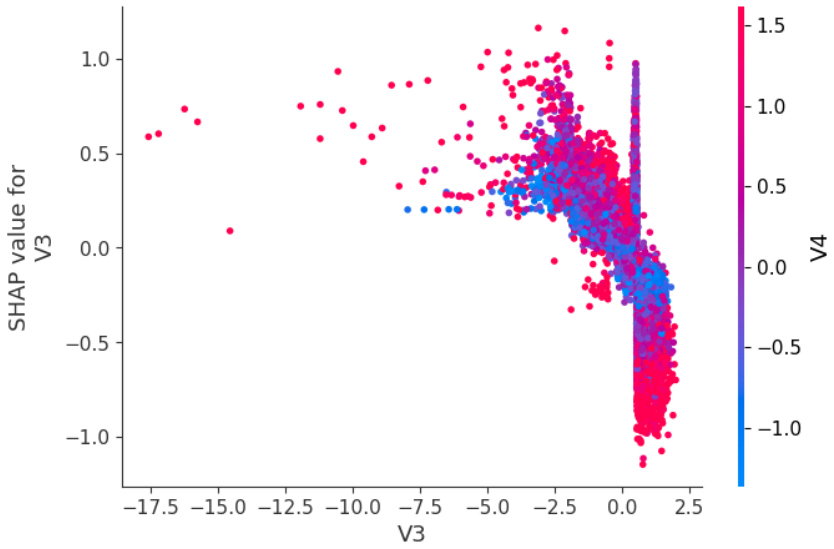


Figure 16: Dependence Plot for V3

[10]SHAP documentation, last checked 17/07/2024 on *https : //shap − lrjball.readthedocs.io/en/latest/example_notebooks/plots/dependence_plot.html*

6 Considering Costs

The context of fraud detection is difficult to address because the errors made by the model bring a direct financial cost. Additionally, the financial impact of these errors varies significantly depending on the type of error, whether it involves a missed fraud (False Negative) or a mistake involving legitimate transactions (False Positive). Each of these costs also differs depending on the transfer itself, as well as other factors intrinsically linked to the transfer such as the bank, location, date, etc. For this reason, we have studied several possibilities to allow our models to understand the costs associated with these errors.

6.1 Cost Weighting Approach

Our first idea was to use the non-anonymized variable representing the transaction amounts as weights in the model's decision-making process. Indeed, if we look at the results provided in Table 7, we can see that although the performances seem slightly lower than our best model, the cost of errors is lower. We deduce that the model ends up making more errors, but those errors are less costly.

Model	**AUC**	**Accuracy**	**F1**	**Precision**	**Recall**	**PR AUC**	**Cost**
LightGBM-BEST	0.937	0.999	0.915	0.984	0.776	0.864	2781.19
LightGBM-Cost	0.937	0.999	0.855	0.951	0.776	0.864	2480.28

Table 7: Model performance with cost instance weighting

Although this method seems effective at first glance, it is important to remember that the cost of an undetected fraud or a wrongful accusation on a legitimate transfer is generally not the amount of the transfer. It is indeed a good idea because the larger the amount of the transfer, the greater the costs incurred by an undetected fraud. Moreover, the financial impact of a wrongful accusation can match the scale of the transfer amount.[11]

6.2 Training with Adapted Metrics

Our second idea was to consider training the model when we know the exact cost. If we allow the user to input the costs considered in this context, we could train a model that aims to minimize these given costs. It is important to specify that here we are talking about estimated average costs (because it is difficult to obtain the exact costs generated by an error due to costs sometimes being related to the transfer itself, see 4.2).

Therefore, we demonstrate at point 4.2 that using the F_β score with $\beta^2 = \frac{C_{fn}}{C_{fp}}$ is beneficial. Assuming that the cost of a wrongful accusation is 100 times more costly than a fraud, we can search for the optimal model.

[11] Acknowledgment to Arthur Desaive, a graduate student of the University of Liège in the Faculty of Law.

However, from a usability standpoint, performing tuning to create a new model adapted to the costs incurred by errors involves latency on the application during the time it takes to find the optimal hyperparameters for the newly defined F_β score and train this new model. As a result, this approach may not be feasible as the data volume increases and thus the training time extends.

6.3 Thresholding

The last approach considered was thresholding. By default, models assign a threshold of 0.5 for the final decision. This means that the model calculates the probability of each instance being fraud or not, and if this probability is greater than 0.5, the classifier will classify the transfer as fraudulent and vice versa. However, it is possible to modify this threshold so that the model tends to classify more instances as fraud or vice versa. Our final approach is to find the optimal threshold that minimizes the average cost provided by the user. The great advantage of this method is that it does not require retraining the model and is thus not as demanding in terms of computation time.

First, we defined a function to find the best threshold that minimizes a cost function we also created. The idea behind this approach is to consider 40 thresholds between 0 and 1 and find the optimal threshold based on costs. To do this, we divide our training set into 3 stratified datasets and alternately train our model. For each model, we calculate the cost function:[12]

$$J(y, \hat{y}) = C_{fp} \times (1 - \hat{y}) \times y + C_{fn} \times \hat{y} \times (1 - y)$$

We then retrieve the average cost of these models based on the considered threshold. In the end, we simply return the threshold allowing for the lowest cost. Our experimental results prove that when the cost of not detecting a fraud increases and surpasses the cost of a wrongful accusation, the threshold will be adjusted to find the optimal cost. For example, if the cost of an undetected fraud becomes 100 times greater than the cost of a wrongful accusation, we obtain an optimal threshold of 0.076, which gives us a cost of 510 compared to 1101 for our model optimized according to precision.

On May 27, 2024, a new function from the Scikit-learn library named TunedThresholdClassifierCV was created. This function post-tunes the decision threshold using cross-validation. When using this function, we aimed to optimize the F_β score with $\beta^2 = \frac{C_{fn}}{C_{fp}}$. Although the two methods seem similar, the method we developed and the one implemented in TunedThresholdClassifierCV did not provide the same results. Indeed, through experimental tests, we found that postprocessing with TunedThresholdClassifierCV did not provide an optimal cost. We propose several hypotheses for this result:

1. The TunedThresholdClassifierCV function may have been misused on our part.
2. The F_β score does not minimize costs as we thought, and thus the result obtained at point 4.2 may not be correct.
3. The TunedThresholdClassifierCV function contains errors.

[12] Fraud detection with cost-sensitive machine learning by Roman Moser in Towards Data Science, 29/03/2019. Last check: 10/07/2024. https://towardsdatascience.com/fraud-detection-with-cost-sensitive-machine-learning-24b8760d35d9

For these various reasons and because our experiments indicate that the thresholding method used in this project works, we have retained our method despite it being more time-consuming in terms of computation.

Although the thresholding method seems to provide a concrete solution to our problem, it is worth noting that thresholding may not always lead to the optimal cost for the given costs, which could potentially be achieved through the retraining method by tuning our hyperparameters.

7 Drifting and Continuous Development

According to Intech, the last challenge of this project was to create accessible and performant models over time. This means that the models needed to be easily trainable, capable of being retrained and redeployed without disrupting the application. Each model must therefore be deployed independently and accessible via requests. Additionally, the phenomenon of drifting is highly prevalent in the field of fraud detection. Fraudsters are constantly seeking new ways to reinvent fraud in order to circumvent security measures like our machine learning models. Moreover, the different variables in our dataset represent information about transactions and enable our model to differentiate between usual transactions and fraud. However, the values of usual transactions can vary over time as they may depend on external factors such as purchasing power, the arrival of new banks, new currencies, etc.

7.1 Continuous Development with MLOps

An important aspect of managing drifting and deploying a machine learning model is the MLOps management of the project. This MLOps management relies on several key steps that we aim to automate.[13]

1. Integration of new data: Being able to store a data flow. In our case, we used Apache Kafka[14], which according to Intech, is the most appropriate technology to use and is already utilized within the company.
2. Preparation, storage, and versioning of data: Once the data is integrated, we can perform the preprocessing defined in Section 3.1. For storing this data, we used a technology called MinIO[15], which is the most used storage in the field of machine learning and is widely used within Intech.
3. Retraining the model: Determine if model retraining is necessary, and if so, retrain it, version it, and store it on a storage platform. Here, we will also use MinIO to store our models.
4. Adapting the model as a REST API: We used FAST API to generate these APIs for the various requests necessary for the application (prediction, explanation, evaluation, etc.).
5. Dockerization of the model: The model must be transformed into an image (here a Docker image) that can be pushed and pulled from a registry (here we use Harbor technology, which provides an internal registry to Intech).
6. Deployment and Orchestration of Docker: The final point involves orchestrating these different Docker containers using an orchestrator such as Kubernetes.

[13] Réentrainer automatiquement un modèle : fausse bonne idée ?, Paul PETON, 18/12/2021, Last checked 15/07/2024, https://paul-peton.medium.com/réentrainer-automatiquement-un-modèle-fausse-bonne-idée-a885e0783d86

[14] Apache Kafka Documentation, Last checked on 19/08/2024 on https://kafka.apache.org/documentation/

[15] Minio documentation, Last checked on 19/08/2024 on https://min.io/docs/minio/kubernetes/upstream/

7.2 Deployment of Models

It is important to revisit the deployment part of the models mentioned earlier. To effectively improve and update our application, we opted for a microservices approach as the deployment structure of our project.

The microservices approach is an architectural style where an application is decomposed into small and independent services, called microservices.[16] Each microservice is designed to perform one or more specific tasks unique to that microservice. In our case, we will have one microservice per model and one microservice for our application. Our application will allow access to information from our models via API requests defined using FAST API. This architecture allows for model updates without affecting the rest of the application. It also makes it easier to roll back if we find that a new model does not function properly or makes too many errors.

The microservices architecture of our project includes several models where each model runs in a Docker container. These Docker containers are orchestrated within a Kubernetes cluster. In this cluster, each Docker container runs in a pod, managed by a deployment (which allows for scaling, ensuring the desired number of running pods, etc.) and accessible via a service. The application itself has an additional LoadBalancer service linked to an Ingress, making it accessible to an external user. More information about the application can be found on Section A in the appendix. The schema of our Kubernetes cluster architecture can be found in Figure 20 while the schema of the all architecture of the deployment use in this project can be found on Figure 19.

7.3 Concept Drift

Concept drift is defined as:

> *"Concept drift describes unforeseeable changes in the underlying distribution of streaming data over time. Data analysis has revealed that machine learning in a concept drift environment will result in poor learning results if the drift is not addressed"*[44]

As Jie Lu mentioned in his paper, the problem of drift can reduce the performance of our models if not managed. This phenomenon, also known as data shift or concept shift, can occur for several reasons:

1. **Model drift:** This type of drift occurs when the task for which the model is designed changes completely. In our case, this corresponds to the emergence of a new type of fraud or the disappearance of an existing one. These new fraud types may not be detected by the model.
2. **Data drift:** This type of drift happens when the distribution of variables changes, making values that were previously considered anomalies by the model now appear normal due to shifts in the value distributions over time. A good example would be a change in the distribution of a monetary value. For instance, if the value of the euro were to drop significantly, the distribution of transaction values could change and show much higher values.

[16] Microservice Architecture pattern, Chris Richardson, Last checked 15/07/2024 on https://microservices.io/patterns/microservices.html

7.4 Handling Drift

There are several methods to address the concept of drift, including approaches related to model performance or data distribution [44].[17]

1. **Performance Approach:**

   This simplistic approach retrains the model when its performance begins to decline.

2. **Distribution Approach:**

   This method estimates the distance between the distributions of the historical dataset on which the model was trained and the new dataset that has just been relabeled. To calculate these distances, we can use methods such as Kolmogorov-Smirnov, Jensen-Shannon, or Wasserstein.

3. **Population Stability Index (PSI):**

   The PSI is a statistical measure used to compare the distribution of a categorical variable across two different datasets. High PSI values indicate a significant difference between the distributions of the variable in the two datasets.

4. **Page-Hinkley Method:**

   The Page-Hinkley method works by accumulating the difference between observed data points and a reference value (typically the mean of the initial data). A significant increase in the cumulative sum of these differences indicates a potential model drift.

7.5 Retraining Models

When our models are no longer up to date, it is necessary to retrain them. However, there are several retraining options:

1. **Complete Retraining:**

   The most basic method is to simply reselect part or all of the historical dataset along with the new dataset to create a new model. However, this method can quickly become time-consuming (as the amount of data increases) and is not always practical in real cases.

2. **Incremental Retraining:**[18]

   Some models train iteratively on data batches, making it possible to retrain them incrementally on the new dataset without completely retraining the model. This method is less time-consuming and allows for frequent and quick updates. However, over time, this method can introduce bias. Additionally, it is only applicable to machine learning models that support incremental learning.

3. **Transfer Learning:**[19]

[17] Understanding Data Drift and Model Drift, Moez Ali, 01/2023, Last checked 15/07/2024 on https://www.datacamp.com/tutorial/understanding-data-drift-model-drift

[18] What is Incremental Learning?, Abid Ali Awan, 06/2023, last checked 15/07/2024 on https://www.datacamp.com/blog/what-is-incremental-learning

[19] What is transfer learning?, Jacob Murel, Eda Kavlakoglu, 12/02/2024, last checked 15/07/2024 on https://www.ibm.com/topics/transfer-learning

Transfer learning is a technique in which knowledge gained in one context is used to improve generalization in another context. This method reduces training time and the dataset required for training. It also allows for better generalization by using information learned on an older dataset while incorporating new information. However, for transfer learning to be effective, the following three conditions must be met:

(a) First, the model's task must not change (i.e., the criteria for classifying fraud should not change completely, which can happen when a new type of fraud emerges).

(b) The distributions of the new dataset and the old dataset should not vary too much.

(c) The historical model and the model to be trained should be comparable.

7.6 Our Approach

Once we obtain a new dataset relabeled by an expert who may have corrected the model's errors, we start by testing if the model's prediction on this new dataset yields results lower than expected in terms of F1 score. Specifically, we test on the new dataset and the historical dataset; if the results are lower, we might be experiencing drift. We then conduct a probabilistic distance test between the new data and the historical data. If this distance exceeds a certain threshold, we perform a complete retraining of our model; otherwise, we perform transfer learning retraining of our model.

This method has been developed in the file pipeline.py for LightGBM. However, we were unable to implement this same pipeline for other models due to time constraints.

All the architecture of our platform can be found on Figure 19 while the architectue of the kubernetes cluster can be found on Figure 20.

7.7 Kubeflow

Finally, we considerate a purely MLOps approach. To optimize the retraining of models, we sought technologies that could facilitate this process.

In our research on how to optimize the retraining system for our models, we reviewed numerous topics on MLflow and Kubeflow. Although these two technologies offer different tools within the MLOps framework, we decided to focus more on Kubeflow. This tool was designed to be used as an overlay on a Kubernetes cluster, whereas MLflow can be used with other orchestrators. Among its many features, Kubeflow includes three major tools that seemed useful for our project: Katib, KServe, and Kubeflow Pipelines.

7.7.1 Katib

Katib is a Kubeflow feature that enables automated machine learning. It can be used to effectively tune the hyperparameters of our models. This technology allows for scaling and parallelizing the different steps of hyperparameter tuning by running them on different Docker containers.

7.7.2 Kubeflow Pipelines

Kubeflow Pipelines is a platform for creating machine learning pipelines with a user interface that makes it easier to manage and track experiments, jobs, and runs. Directly associated with a notebook, it is possible to create our pipelines directly on Kubeflow. A pipeline represents the various steps in performing machine learning training. Each of these steps is deployed in a Docker container. The various steps include:

- Downloading the dataset
- Scaling the data
- Handling imbalanced data
- Hyperparameter tuning (with Katib, for example)
- Training the model
- Evaluating the model on the test set

Each of these steps is thus dockerized; in the context of Kubeflow Pipelines, each Docker container is called a component. Each component is reusable. This allows us to retrain the model without retraining certain preprocessing steps by reusing the preprocessing components and only changing the training component. Moreover, creating the pipeline is considered as a Kubernetes job. It is easy to create cronjobs (also called recurring runs in Kubeflow) to run the pipeline job at predefined time intervals. It is also possible to create conditional components, meaning we can execute a specific component only if a condition is met (e.g., retraining the model only if drifting is detected). Finally, the interface provides a visual representation of our entire machine learning infrastructure and shows the various results of our models. An example of a Kubeflow Pipeline is shown in Figure 17. In this figure, all the learning parts are inside the conditional branching components, which represent the conditional choice of whether or not to train our model based on a potential drift.

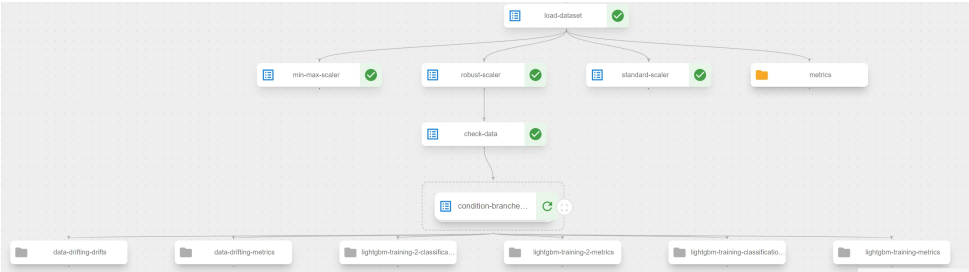


Figure 17: Kubeflow Pipeline for LigthGBM training

7.7.3 KServe

The last tool from Kubeflow is KServe. KServe is a standard Model Inference Platform on Kubernetes, built for highly scalable use cases. It allows for advanced deployment methods such as canary rollouts. It integrates the use of explainable methods like SHAP. However, this method will not be used in our thesis for several reasons. First, model inference is limited to the options provided by KServe, which are sometimes too limited for specific uses needed in our application and generally more complex to implement

than simply creating an API. Moreover, an inference method had already been developed using FAST API before considering the option of using Kubeflow in this project. Finally, KServe is available for a limited number of models. For instance, it is not possible to use this method for serving Explainable Boosting or CatBoost models.

For all these reasons, we decided not to use this tool in our thesis.

Finally, other Kubeflow tools, such as Spark Operator, Models Registry, and Training Operator, seem to be very useful. However, they were not considered in this thesis mainly due to time constraints or because they were perhaps too rigorous and demanding in terms of time investment compared to their utility (Models Registry, for example).

8 Discussion

In this chapter, we will discuss the various results obtained in this thesis. These results will be divided according to the objectives that were predefined in the introduction of this report in Section 1.2. For each issue raised, we will discuss what was addressed in relation to that objective.

8.1 Model Development in Imbalanced Situations

We provided a novel approach using WCGANs [12], which gave us very satisfactory results with an F1 score of 0.91 with new models such as Extreme Gradient Boosting [40].Our approach is novel since GAN models have not been fully explored in the field of fraud detection, and the WCGAN, in particular, does not appear to have ever been used in this context. Additionally, we explored new ensemble models like CatBoost [42] and Light Gradient Boosting [41], achieving results up to 0.92 in F1 score when we adopted a cost-sensitive approach. All these results are significantly better than what had been previously achieved with the same dataset, as we have recorded in Table 1.

8.2 Model Development in the Context of Interpretability

We adopted an methodological approach to analyze and utilize the explanatory method that is the simplest (i.e., least computationally expensive) and the most truthful (i.e., where we can almost completely explain the process behind the explanation). In this research, we found that SHAP [34] was the technology that best met these criteria.

Next, we addressed interpretable models, seeking a model that could be fully interpretable while only losing a little efficiency in the results. In this research, the Explainable Boosting Model [43] using the GA2M [32] method seemed to be the most suitable model to best resolve these issues, achieving results up to an F1 score of 0.9.

8.3 Continuous Training and Application Deployment

We aimed to develop an application that detects fraudulent transactions rather than just models. This application needed to be deployed and sustainable over time. To achieve this, we used Kubernetes[20] to deploy our various models, as well as Kubeflow[21] to ensure continuous retraining of our models when faced with model drifting or data drift. This method seems to work well with our artificial tests; however, due to factors beyond our control, we were unable to test it in a real-world situation.

8.4 Approach Related to Monetary Costs

We explored three possibilities to account for the monetary values behind potential model errors. This approach was considered to enable direct discussions with future Intech clients about their actual situations and to determine the most practical method for their specific case. The three possibilities explored were:

- Instance weighting with the direct cost of each transaction.

[20]Kubernetes documentation, Last checked on 12/08/2024 on https://kubernetes.io/
[21]Kubeflow documentation, Last checked on 12/08/2024 on https://www.kubeflow.org/

- Fine-tuning of hyperparameters on a cost-adapted metric (F_β score).
- Thresholding of our models based on the average costs attributed to errors.

Each of these methods addresses the problem in different ways, allowing for different approaches to each client's specific issues.

9 Limitation

In this chapter, we will present the various limitations we encountered during this thesis.

9.1 Datasets

The first limitation was, of course, the difficulty in obtaining a dataset. The internal restructuring of POST Group made exchanges between Intech and their banking intermediary, BGL, difficult. These challenging exchanges complicated the signing of contracts allowing access to such data, which is protected by GDPR. Additionally, client-company communication was not optimal for the same reasons.

As a result, we used the dataset provided by the Université Libre de Bruxelles. However, this dataset is quite limited (only 470,000 transactions) and contains a higher presence of fraud compared to reality, according to Intech. Moreover, this dataset is anonymized via the PCA method, leading to the loss of much information about the data in the process. Additionally, this PCA transformation prevents us from performing statistical or machine learning methods that could be useful, such as clustering based on categorical variables. For example, clustering by country could be helpful since a transaction from Bangladesh is logically more likely to be fraudulent than a transaction from Belgium.

The small amount of data prevents us from using the latest deep learning models (our best MLP achieved around 0.86 in F1 score), so a separate study would be useful to consider the option of different deep learning models.

9.2 Time Constraints

The duration of this master's thesis being six months does not leave much time to explore the entirety of the existing methods. For example, as we saw in Section 3.2, WCGAN did not always provide distributions close to our data. Some other GANs, such as CTGAN [45], seem to find distributions closer to our data.

Moreover, the time allotted for this project did not allow for a full exploration of the Kubeflow technology. We were able to achieve continuous training for one model, but due to time constraints, we could not explore additional solutions for all the models considered in this thesis. Other MLOps technologies, such as Rflow or MLflow, were not explored. Moreover, since the continuous training aspect of the project was the last to be considered, due to a lack of time, there was no opportunity to conduct a thorough state-of-the-art review or explore research in this area.

Finally, with more time, we could have completed other steps in our fraud detection process. Indeed, our platform does not include all the possibilities proposed by Dal Pozzolo [6]. For example, a model evaluating the risk of fraud upstream and the implementation of predefined rules would increase the chances of detecting larger frauds.

9.3 Libraries in Development

Many of the technologies we used are evolving rapidly, with new versions released frequently. For example, InterpretML is currently considering a cost-sensitive approach for

their models and is working on implementing it in their library. Additionally, Imbalanced-learn has released several functions for thresholding, which we did not have time to explore.

Similarly, we used version 1.8 of Kubeflow to ensure stability in our application. However, version 1.10 is currently in development, with new features for Katib to fine-tune the hyperparameters of our models in the best way possible.

10 Conclusions and Future work

In conclusion, this project successfully developed a fraud detection platform deployed using various frameworks. The research behind this platform led to improved performance by leveraging newer models, such as Light Gradient Boosting and Extreme Gradient Boosting, which are not yet widely used in the fraud detection field. We explored different solutions for handling data imbalance through cost-sensitive learning or sampling via the Wasserstein Conditional Generative Adversarial Network (WCGAN). We also addressed the concept of explainability using two options: SHAP and Explainable Boosting. Subsequently, we devised three methods to account for the monetary cost of errors: Direct Weighting, Fine Tuning, and Thresholding. Finally, we developed a solution to ensure our models are resistant to data and model drift using Kubeflow.

The impact of this thesis has been to introduce new approaches in the field of fraud detection, leading to better results than those found in the existing literature. For Intech, this thesis also provided valuable insights into the potential impact of Kubeflow in the broader world of machine learning. Additionally, this thesis demonstrates to potential future clients what Intech is capable of in terms of machine learning model development. Furthermore, if Post desires, this project could serve as the foundation for developing a more comprehensive and complex fraud detection platform. Lastly, this work has helped to identify the available options for incorporating monetary costs into fraud detection models.

In continuation of this thesis, there are many opportunities for future work. Indeed, this thesis is just a step toward what is possible in the field of fraud detection.

First, it may be possible to conduct a comparative study between different GANs to understand why WCTGAN is used in fraud detection compared to others such as CTGAN in order to find better data generators, which is a crucial aspect of fraud detection.

Next, conducting research that evaluates the potential of using Graph Neural Networks (GNNs) could yield promising results, as proposed by Wei Zhuo [7]. Furthermore, research already exists to make these Graph Neural Networks explainable [46]. GNNs are rapidly growing in the field of deep learning. If this type of model proves effective, it might be useful to consider a new version of fraud detection system design tailored to this model (with rules specific to graphs, for example).

Additionally, with a larger dataset, deep learning methods could be employed to predict whether a transaction is fraudulent. These deep learning methods can be interpreted using techniques like Neural Additive Models [47]. This approach complicates the principle of GAMs by incorporating Neural Networks.

Subsequently, with direct contact with industry experts, several adaptations could be made based on their actual needs. Additionally, with more information on the data (and therefore less anonymization), it would be possible to highlight the actual variable values through the explainability of our models and thus provide real context behind these explanations. For example, one could identify countries where fraud is more prevalent or types of transactions that have a higher probability of fraud. Based on these results and further discussions with experts, certain rules could be redefined, making it easier to predict fraud.

Moreover, Explainable Boosting is a model that already seems highly effective in the do-

main of interpretable models. However, as explained in Section 4.5, there is currently no method to make the model cost-sensitive. A detailed study of the model and a deeper examination of the algorithm's code could enable the implementation of such learning. This model could then show even more impressive results in the field of fraud detection.

Furthermore, if a large amount of data is accumulated, it could be possible, using increasingly powerful new LLM models, to analyze transactions as a set of characteristics rather than tabular information, thereby learning to differentiate frauds from non-frauds. Models such as LAMA 3, recently released by Meta, could be tested to see if this type of model could be effective in the field of fraud detection.

Finally, new technologies are emerging, and recently, we have heard about KAN (Kolmogorov–Arnold Networks) [48]. The main idea behind this new technology is to have learnable activation functions rather than fixed ones and to use only sum operations on the nodes. These new models could offer better accuracy while maintaining model explainability. Following this project, it might be useful to test this new model in an imbalanced context like fraud detection and see if the imbalance management methods proposed in this thesis also work well with these models.

References

[1] P. Geurts and L. Wehenkel. *Introduction to Machine Learning*. Course ID: ELEN0062-1. Course. 2021-2022. URL: https://www.programmes.uliege.be/archives/20212022/cocoon/cours/ELEN0062-1.html (pages 8, 29).

[2] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. "Credit card fraud detection and concept-drift adaptation with delayed supervised information". In: *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*. IEEE, 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280527. URL: https://doi.org/10.1109/IJCNN.2015.7280527 (pages 10, 11).

[3] Eunji Kim, Jehyuk Lee, Hunsik Shin, Hoseong Yang, Sungzoon Cho, Seung-kwan Nam, Youngmi Song, Jeong-a Yoon, and Jong-il Kim. "Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning". In: *Expert Syst. Appl.* 128 (2019), pp. 214–224. DOI: 10.1016/J.ESWA.2019.03.042. URL: https://doi.org/10.1016/j.eswa.2019.03.042 (pages 10, 11).

[4] Wesley Kenneth Wilhelm. "The Fraud Management Lifecycle Theory: A Holistic Approach to Fraud Management." In: 2004. URL: https://api.semanticscholar.org/CorpusID:1828823 (page 10).

[5] Anna Nesvijevskaia, Sophie Ouillade, Pauline Guilmin, and Jean-Daniel Zucker. "The accuracy versus interpretability trade-off in fraud detection model". In: *Data 38; Policy* 3 (2021), e12. DOI: 10.1017/dap.2021.3 (pages 10, 11).

[6] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy". In: *IEEE Trans. Neural Networks Learn. Syst.* 29.8 (2018), pp. 3784–3797. DOI: 10.1109/TNNLS.2017.2736643. URL: https://doi.org/10.1109/TNNLS.2017.2736643 (pages 11, 61).

[7] Wei Zhuo, Zemin Liu, Bryan Hooi, Bingsheng He, Guang Tan, Rizal Fathony, and Jia Chen. "Partitioning Message Passing for Graph Fraud Detection". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=tEgrUrUuwA (pages 11, 63).

[8] Tungyu Wu and Youting Wang. "Locally Interpretable One-Class Anomaly Detection for Credit Card Fraud Detection". In: *CoRR* abs/2108.02501 (2021). arXiv: 2108.02501. URL: https://arxiv.org/abs/2108.02501 (page 11).

[9] Hao An, Ruotong Ma, Yuhan Yan, Tailai Chen, Yuchen Zhao, Pan Li, Jifeng Li, Xinyue Wang, Dongchen Fan, and Chunli Lv. "Finsformer: A Novel Approach to Detecting Financial Attacks Using Transformer and Cluster-Attention". In: *Applied Sciences* 14.1 (2024). ISSN: 2076-3417. DOI: 10.3390/app14010460. URL: https://www.mdpi.com/2076-3417/14/1/460 (page 11).

[10] Guillaume Lemaitre, Fernando Nogueira, and Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *J. Mach. Learn. Res.* 18 (2017), 17:1–17:5. URL: http://jmlr.org/papers/v18/16-365.html (pages 12, 15).

[11] Peter Gnip, Liberios Vokorokos, and Peter Drotár. "Selective oversampling approach for strongly imbalanced data". In: *PeerJ Comput. Sci.* 7 (2021), e604. DOI: 10.7717/PEERJ-CS.604. URL: https://doi.org/10.7717/peerj-cs.604 (pages 12, 13).

[12] Justin Engelmann and Stefan Lessmann. "Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning". In: *Expert Syst. Appl.* 174 (2021), p. 114582. DOI: 10.1016/J.ESWA.2021.114582. URL: https://doi.org/10.1016/j.eswa.2021.114582 (pages 12, 13, 26, 27, 59).

[13] Zhuoyuan Zheng, Yunpeng Cai, and Ye Li. "Oversampling Method for Imbalanced Classification". In: *Comput. Informatics* 34.5 (2015), pp. 1017–1037. URL: http://www.cai.sk/ojs/index.php/cai/article/view/1277 (page 14).

[14] Peter E. Hart. "The condensed nearest neighbor rule (Corresp.)" In: *IEEE Trans. Inf. Theory* 14.3 (1968), pp. 515–516. DOI: 10.1109/TIT.1968.1054155. URL: https://doi.org/10.1109/TIT.1968.1054155 (page 14).

[15] Ivan Tomek. "Two modifications of CNN". In: *IEEE Trans. Syst. Man Cybern.* 6.11 (1976), pp. 769–772. DOI: 10.1109/TSMC.1976.4309452 (page 14).

[16] Show-Jane Yen and Yue-Shi Lee. "Cluster-based under-sampling approaches for imbalanced data distributions". In: *Expert Syst. Appl.* 36.3 (2009), pp. 5718–5727. DOI: 10.1016/J.ESWA.2008.06.108. URL: https://doi.org/10.1016/j.eswa.2008.06.108 (page 14).

[17] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance". In: *IEEE Trans. Syst. Man Cybern. Part A* 40.1 (2010), pp. 185–197. DOI: 10.1109/TSMCA.2009.2029559. URL: https://doi.org/10.1109/TSMCA.2009.2029559 (page 15).

[18] Debashree Devi, Saroj K. Biswas, and Biswajit Purkayastha. "A Review on Solution to Class Imbalance Problem: Undersampling Approaches". In: *2020 International Conference on Computational Performance Evaluation (ComPE)*. 2020, pp. 626–631. DOI: 10.1109/ComPE49325.2020.9200087 (page 15).

[19] Pedro M. Domingos. "MetaCost: A General Method for Making Classifiers Cost-Sensitive". In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*. Ed. by Usama M. Fayyad, Surajit Chaudhuri, and David Madigan. ACM, 1999, pp. 155–164. DOI: 10.1145/312129.312220. URL: https://doi.org/10.1145/312129.312220 (page 15).

[20] Kai Ming Ting. "An Instance-Weighting Method to Induce Cost-Sensitive Trees". In: *IEEE Trans. Knowl. Data Eng.* 14.3 (2002), pp. 659–665. DOI: 10.1109/TKDE.2002.1000348. URL: https://doi.org/10.1109/TKDE.2002.1000348 (page 15).

[21] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. "Cost-Sensitive Learning". In: *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing, 2018, pp. 63–78. ISBN: 978-3-319-98074-4. DOI: 10.1007/978-3-319-98074-4_4. URL: https://doi.org/10.1007/978-3-319-98074-4_4 (page 15).

[22] Charles X. Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. "Decision trees with minimal costs". In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. Ed. by Carla E. Brodley. Vol. 69. ACM International Conference Proceeding Series. ACM, 2004. DOI: 10.1145/1015330.1015369. URL: https://doi.org/10.1145/1015330.1015369 (page 16).

[23] Victor S. Sheng and Charles X. Ling. "Thresholding for Making Classifiers Cost-sensitive". In: *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Con-*

ference, July 16-20, 2006, Boston, Massachusetts, USA. AAAI Press, 2006, pp. 476–481. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-076.php (page 16).

[24] Emmanuel Ileberi, Yanxia Sun, and Zenghui Wang. "A machine learning based credit card fraud detection using the GA algorithm for feature selection". In: *J. Big Data* 9.1 (2022), p. 24. DOI: 10.1186/S40537-022-00573-8. URL: https://doi.org/10.1186/s40537-022-00573-8 (pages 16, 28).

[25] Siddhant Bagga, Anish Goyal, Namita Gupta, and Arvind Goyal. "Credit Card Fraud Detection using Pipeling and Ensemble Learning". In: *Procedia Computer Science* 173 (2020). International Conference on Smart Sustainable Intelligent Computing and Applications under ICITETM2020, pp. 104–112. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.06.014. URL: https://www.sciencedirect.com/science/article/pii/S1877050920315167 (page 16).

[26] Noor Alfaiz and Suliman Fati. "Enhanced Credit Card Fraud Detection Model Using Machine Learning". In: *Electronics* 11 (Feb. 2022), p. 662. DOI: 10.3390/electronics11040662 (page 17).

[27] Abdul Rehman Khalid, Nsikak Pius Owoh, Omair Uthmani, Moses Ashawa, Jude Osamor, and John Adejoh. "Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach". In: *Big Data Cogn. Comput.* 8.1 (2024), p. 6. DOI: 10.3390/BDCC8010006. URL: https://doi.org/10.3390/bdcc8010006 (pages 17, 26).

[28] Finale Doshi-Velez and Been Kim. "A Roadmap for a Rigorous Science of Interpretability". In: *CoRR* abs/1702.08608 (2017). arXiv: 1702.08608. URL: http://arxiv.org/abs/1702.08608 (page 18).

[29] Sajid Ali, Tamer Abuhmed, Shaker H. Ali El-Sappagh, Khan Muhammad, Jose Maria Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz Rodríguez, and Francisco Herrera. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence". In: *Inf. Fusion* 99 (2023), p. 101805. DOI: 10.1016/J.INFFUS.2023.101805. URL: https://doi.org/10.1016/j.inffus.2023.101805 (page 18).

[30] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable AI: A Review of Machine Learning Interpretability Methods". In: *Entropy* 23.1 (2021), p. 18. DOI: 10.3390/E23010018. URL: https://doi.org/10.3390/e23010018 (pages 18, 19).

[31] Berk Ustun and Cynthia Rudin. "Supersparse linear integer models for optimized medical scoring systems". In: *Mach. Learn.* 102.3 (2016), pp. 349–391. DOI: 10.1007/S10994-015-5528-6. URL: https://doi.org/10.1007/s10994-015-5528-6 (page 19).

[32] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. "Accurate intelligible models with pairwise interactions". In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. Ed. by Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy. ACM, 2013, pp. 623–631. DOI: 10.1145/2487575.2487579. URL: https://doi.org/10.1145/2487575.2487579 (pages 20, 30, 59).

[33] L. S. Shapley. "17. A Value for n-Person Games". In: *Contributions to the Theory of Games (AM-28), Volume II*. Ed. by Harold William Kuhn and Albert William Tucker. Princeton: Princeton University Press, 1953, pp. 307–318. ISBN: 9781400881970.

DOI: doi:10.1515/9781400881970-018. URL: https://doi.org/10.1515/9781400881970-018 (page 21).

[34] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 4765–4774. URL: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html (pages 22, 23, 59).

[35] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi. ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778 (page 23).

[36] Muhammad Rehman Zafar and Naimul Mefraz Khan. "DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems". In: *CoRR* abs/1906.10263 (2019). arXiv: 1906.10263. URL: http://arxiv.org/abs/1906.10263 (page 24).

[37] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. "Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods". In: *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*. Ed. by Annette N. Markham, Julia Powles, Toby Walsh, and Anne L. Washington. ACM, 2020, pp. 180–186. DOI: 10.1145/3375627.3375830. URL: https://doi.org/10.1145/3375627.3375830 (page 24).

[38] Elisha Blessing and Hubert Klaus. "Anomaly Detection Techniques to identify outliers or anomalies in datasets, which could be indicative of errors or noteworthy events." In: 3481 (Dec. 2023), p. 18 (page 25).

[39] Maxime Fays. *Machine Learning in Space Sciences*. Course ID: SPAT0263-1. 2023-2024. URL: https://www.programmes.uliege.be/cocoon/20232024/cours/SPAT0263-1.html (page 29).

[40] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi. ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785 (pages 29, 59).

[41] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 3146–3154. URL: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (pages 29, 59).

[42] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. "CatBoost: unbiased boosting with categorical fea-

tures". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 6639–6649. URL: https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285Abstract.html (pages 29, 59).

[43] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. "InterpretML: A Unified Framework for Machine Learning Interpretability". In: *CoRR* abs/1909.09223 (2019). arXiv: 1909.09223. URL: http://arxiv.org/abs/1909.09223 (pages 30, 31, 40, 59).

[44] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. "Learning under Concept Drift: A Review". In: *CoRR* abs/2004.05785 (2020). arXiv: 2004.05785. URL: https://arxiv.org/abs/2004.05785 (pages 54, 55).

[45] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. "Modeling Tabular data using Conditional GAN". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 7333–7343. URL: https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html (page 61).

[46] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in Graph Neural Networks: A Taxonomic Survey". In: *CoRR* abs/2012.15445 (2020). arXiv: 2012.15445. URL: https://arxiv.org/abs/2012.15445 (page 63).

[47] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Benjamin J. Lengerich, Rich Caruana, and Geoffrey E. Hinton. "Neural Additive Models: Interpretable Machine Learning with Neural Nets". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan. 2021, pp. 4699–4711. URL: https://proceedings.neurips.cc/paper/2021/hash/251bd0442dfcc53b5a761e050f8022b8-Abstract.html (page 63).

[48] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. "KAN: Kolmogorov-Arnold Networks". In: *CoRR* abs/2404.19756 (2024). DOI: 10.48550/ARXIV.2404.19756. arXiv: 2404.19756. URL: https://doi.org/10.48550/arXiv.2404.19756 (page 64).