One Step Ahead

A weakly-supervised approach to training robust machine learning models for transaction monitoring

by

D.J. van der Werf

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 26 November 2021 at 09:00 AM.

Student number:	4369556	
Project duration:	November 1, 2020 – November 26, 2021	
Thesis committee:	Dr. ir. J. Yang,	Tu Delft, Assistant Professor and Thesis Supervisor
	Prof. dr. ir. G.J.P.M Houben,	Tu Delft, Full Professor
	Dr. ir. L. Cavalcante Sieb,	Tu Delft, Assistant Professor
	Ir. A.M.A Balayn,	Tu Delft, PhD Student
	Dr. ir. A. El Hassouni,	bunq, Data Lead

TU Delft

Preface

Dear reader, in front of you lies my thesis on the creation of robust weakly-supervised machine learning models for transaction monitoring. In this work a weakly-supervised machine learning pipeline is introduced with the goal of adapting to the challenges posed by the ever-changing field of financial fraud detection. In other words: to be one step ahead of fraudsters. We show that by generating and carefully selecting synthetic data representing fraudulent transactions we are able to train weakly-supervised models that outperform their supervised counterparts on nearly all performance metrics used in the field. The most striking result is a significant increase in terms of the robustness to never-seen-before fraudulent behavior. Our pipeline is shown to correctly classify roughly 7% more fraudulent transactions than baseline supervised models. These instances of fraud would previously go undetected.

This thesis was conducted at the Web Information Systems group at the Delft University of Technology. This work is the final step to obtain my masters degree in Computer Science. For roughly 9 months, I have worked on this thesis at a Dutch bank called bunq, on the front lines of transaction monitoring. During this time, I performed work on both the transaction monitoring system at bunq to familiarize myself with the field, as well as research data science and machine learning methods to substantiate my thesis. One of the hardest challenges to overcome was juggling between the needs of both bunq and the TU Delft to create a thesis project that is both viable to a company like bunq and academically interesting to the university. Along with the Covid-19 pandemic being at its height during the majority of my thesis time, balancing between the two for 45 hours a week was not always an easy task. Because of this, I am very proud with what this thesis has become and the accomplishments that I made.

This work would not have the same without the help of some people. First and foremost, I am thankful to my parents, brothers and housemates who were always there for me during these often stressful and sometimes rough times of the Covid-19 pandemic. I would also like to thank the kind people at bunq who were always more than happy to have a nice chat, help out or provide insights on my work. Finally, I would like to specifically thank Ali El Hassouni for his indispensable feedback and discussions on the topic of (un)supervised learning, Jesper Romers for his patience and supervision at bunq, Fabian van Altena for all of his insights on the topic of financial fraud analysis and Jie Yang and Agathe Balayn for their continuous support on the academic side.

This thesis work is the culmination of all my years at the TU Delft and I am proud to present it to any interested reader. For this reason, this work includes an extensive background research into all aspects of the introduced pipeline to accommodate readers of varying backgrounds. I sincerely hope you will enjoy reading it as much as I enjoyed working on it.

D.J. van der Werf
Delft, November 2021

Contents

Nomenclature

AWS	Amazon Web Services
CAP	Causative Attack Protection (algorithm)
$CTGAN$	Conditional Tabular Generative Adversarial Network
DNB	The Dutch National Bank
ES	ElasticSearch
FDM	Fraud Detection Model
FIU	Financial Intelligence Unit
$FVAM$	Feature value attribution model
GAN	Generative Adversarial Network
GBM	Gradient Boosting Machine
$GDPR$	General Data Protection Regulation
ML	Machine Learning
MVC	Model-View-Controller
PAC	Probably Approximately Correct
PMF	Probability Mass Function
$PPDP$	Privacy Preserving Data Publishing
SDV	Synthetic Data Vault
$SIRA$	Systematic Analysis of Integrity Risks
TMM	Transaction Monitoring Metadata
UU	Unknown Unknown
VGM	Variational Gaussian Mixture model
$Wwft$	Anti-Money Laundering and Anti-Terrorist Financing Act
XGB	eXtreme Gradient Boosting

1 Abstract

In recent years financial fraud has seen substantial growth due to the advent of electronic financial services opening many doors for fraudsters. Consequently, the industry of fraud detection has seen a significant growth in scale, but moves slowly in comparison to the ever-changing nature of fraudulent behavior. As the monetary losses associated with financial fraud continue to grow, so does the need for efficient automated decision making systems. Simple decision making rules are often still the industry standard and only show decent results in the short-term, as reverse-engineering such rules is an easy task for smart fraudsters. Supervised learning systems as automated fraud detectors have shown promising results across the field, but are plagued by challenges uniquely prevalent in the field. Disproportional class imbalance in fraudulent transactions, as well as fraudsters continually adopting new schemes make training robust and generally applicable machine learning models an arduous task. This work introduces a novel machine learning pipeline, which makes use of carefully selected synthetic samples of this minority class to augment the training dataset of the supervised model. Synthetic samples representing fraudulent transactions are filtered based on a novel technique to quantify their expected performance as an adversarial example, using both data-driven and human-expert-driven techniques. By providing the supervised model with high-quality synthetic adversarial examples, we aim to improve its generalizability to never-seen-before fraudulent behavior and, in turn, improve its robustness to the volatile nature of financial fraud. Our results show that weakly-supervised models trained on our augmented datasets are able to detect 7% more fraudulent transactions compared to a baseline model trained on the standard dataset, at the cost of a 1% increase in false positives. Our calculations further show that applying this system could lead to a decrease of 1/6 in monetary losses incurred by financial fraud.

2 Introduction

Over the past decade *financial fraud* has become an increasingly serious and complex problem. Fraudulent schemes have a far-reaching negative impact ranging from funding illicit activities like organised crime or terrorism to widespread loss of reputation of financial institutions. In the Netherlands, a surge in fraudulent schemes such as credit card fraud, corporate fraud and money laundering has caused €49,1 million in damages in 2020 alone [8]. This is an enormous increase of 195% over 2019, where the total amount in damages was €16,6 million. Not only do existing fraudulent methods continue to grow in scale, the advancement in modern technologies in the financial sector opens the door to many new methods of duping an unsuspecting victim out of their money [104]. Furthermore, fraudsters continue to refine their methods, leading to a cat-and-mouse game where anti-fraud measures are always one step behind. As an example, in 2020 a new type of fraud called *numberspoofing*, where the fraudster calls the victim while realistically posing as a renowned financial institution, was responsible for more than 60% of all fraud-related damages caused in the Netherlands [8]. As a result, there is the need for generally applicable fraud detection and prevention measures that are capable of evolving along with fraudulent techniques.

The Dutch National Bank (DNB) requires financial institutions in the Netherlands take measures to prevent such fraudulent activities [23]. They must monitor and analyse unusual patterns in transactions and pay particular attention to customers that carry a higher risk. Failure to do so, even unintentionally, is regarded by the DNB as an economic offense. As a result, large financial institutions spend a significant amount of time and resources in the form of teams of experts to combat fraud. One such financial institution is *bunq: the Bank of The Free*. bunq is a dutch bank that obtained her banking licence in 2015 and continues to grow rapidly. As bunq grows, so does the number of transactions that are processed. The original and most obvious approach to combat fraud is to use human expert auditors to check every transaction for fraudulent behavior. Consequently, as the number of fraudulent transactions grows, so does the number of man-hours needed to check every single transaction. At a certain point the strategy of manually checking transactions becomes far from feasible. This creates the need for large financial institutions to build efficient automated fraud detection systems.

Many *data mining* approaches have seen widespread application and success in related fields such as share market analysis, credit card approval and bankruptcy detection [100]. Most often these approaches are used to find hidden patterns in vast amounts of transaction-related data. In practice however, many (large) financial institutions and banks still rely on more primitive methods to monitor their transactions. Data analysts try to find patterns in historic fraudulent transactions and in turn create simple *monitoring rules* to automatically detect such patterns. The implementation of these rules often constitutes of no more than a few if-statements and thresholds, which are continuously adapted based on the present-day fraud environment. While these rules provide financial institutions with a fairly strong short-term automated decision making tool and thus a reduction in manual labour to a certain degree, they suffer from 2 major flaws: the rules tend to produce a sizeable number of false positives and also adapt very poorly to new situations. Evidently, the field of automated detection and classification of suspicious transactions poses a wide range of challenges and thus we look into the direction of supervised machine learning. Over the past decade, various machine learning techniques to assist decision makers with classifying suspicious transactions have been researched [18] [30] [57].

While these approaches have seen increased application in the field of automated fraud detection, the field has proven to pose various challenges that in many cases significantly hinder the performance of these methods.

The first major challenge is the fact that tabular data representing transactions generally consists of both continuous and discrete columns. On top of this, categorical fields in such a dataset often have a high cardinality and multi-modal, non-gaussian range of discrete values [102]. This heterogeneous nature of tabular data makes it challenging and in some cases impossible to apply certain machine learning frameworks, for instance frameworks relying on linear regression or Bayesian methods [22] [9]. Secondly, fraud classification problems deal with a significant class imbalance. The number of fraudulent transactions is very small compared to the total number of transactions. On top of requiring an immense number of man-hours of human annotators to create a labeled dataset, one of the fundamental problems of supervised learning approaches is that they are known to greatly suffer from class imbalance [71] [42] [42]. Thirdly, as mentioned earlier, the nature of financial fraud is ever-changing. Therefore, the *generalizability* of machine learning models is of utmost importance. One of the root causes of loss of generalizability and another fundamental problem of supervised machine learning models is the model *overfitting* to the training dataset. Although a wide variety of measures exist to reduce overfitting, the context of fraud detection requires us to take extra domain-specific measures. Another, less obvious, cause of the loss of generalizability is the existence of *unknown unknowns* (UUs): high confidence mistakes made by the model. Often, UUs result from a mismatch between the data used for training and the data found after deployment of the model. Causes of this mismatch are underrepresentation or complete absence of certain instances in the training dataset or due to a shift in population, for example when fraudsters adapt their strategy to make the samples in the training dataset no longer accurate [60].

In the context of financial fraud, making classification errors is very costly. To more accurately grasp the impact of these errors we focus on two different types of mistakes: *false positives* and *false negatives*. A false positive in this context means a user wrongly gets classified as a fraudster. This has 2 major consequences: first and foremost, depending on the financial institution, the user might (temporarily) lose access to his or her monetary account and in turn is likely to lose trust in the institution. Furthermore, false positives are costly in terms of the man-hours needed to manually review the flagged transaction. A false negative essentially means that a criminal gets away unnoticed with committing fraud. The consequences of false negatives are, unfortunately, colourful in nature, ranging from providing means for terrorist funding to allowing users to be duped out of large amounts of funds. An ideal machine learning model or pipeline is able to detect a wide variety of fraudulent behavior, while keeping the number of false positives to a minimum. While both types of errors pose egregious risks to both the users and the financial institution itself, relevant literature estimates the costs of false negatives to be significantly higher than that of false positives, mentioning monetary cost ratios ranging from 143:1 to 82:1 [25].

The last concern we consider when working on automated fraud detection is the fact that the training datasets used for fraud classification generally contain a great deal of personal and sensitive information. Even after filtering out identifying information such as user IDs, addresses or names, an adversary with some degree of prior knowledge might still be able to learn identifying information about a user based on a combination of seemingly non-identifying fields [34]. For this reason and more, a financial institution has to take the responsibility of protecting the privacy of her users very seriously. Records containing sensitive data of individuals, such as financial or medical data, needs to be sanitized in order to be securely stored or published. Ideally this sanitized dataset allows for learning valuable information about a population, while allowing nothing to be learned about the individual. Furthermore, data mining algorithms that are trained on data produced by users of a web service are prone to bias resulting from the dataset, which, in the end, is still a product of social behavior. This may lead to automated decision makers overestimating the degree in which certain, often sensitive, attributes impact the decision being made. Offenses such as fraud or intrusion should be inferred from objective misbehavior, rather than being biased towards uncontrollable attributes such as race, gender or religion [47].

In this work we tackle these problems in the field of supervised machine learning for automated fraud detection by introducing a novel pipeline. The first step of this pipeline is the creation of synthetic samples based on the fraudulent samples at our disposal. Being able to produce high-quality synthetic samples is expected to help mitigate the loss in performance due to underrepresentation of the minority class, as well as provide insights into existing ”blind spots” of the supervised model. Techniques ranging from an intricate pre-processing process to custom sampling methods are introduced to construct synthetic fraudulent transactions that are able to accurately capture patterns found in their real counterparts. Then, a filtering method is used to select only those samples from the synthetic dataset that are expected to be a proper adversarial example. In this context an adversarial example is a synthetic sample that represents an instance from a category of fraud that the machine learning model is currently blind to and, in turn, would not be classified as being fraudulent. In other words, it detects those transactions that are UUs to the model. This filtering method creates a score per transaction which aims to quantify how well a sample is expected to perform as an adversarial example. We call this the adversarial success score of the sample.

The adversarial success score is calculated in 3 steps: first, each value of a single transaction based on how unique this value is compared to the average value of a transaction. For numerical features this ”uniqueness” value is based on the average value and standard deviation. For categorical and boolean features this score is based on how unique the feature value is compared to it’s occurrence across the full dataset. Then, a data-driven method that aims to find out which features push the model towards making certain mistakes is applied. We try to quantify the degree in which each feature pushes the model towards making mistakes of the false negative category. In other words: instances where the model miss-classifies fraudulent transactions as being legitimate with a high confidence. To this end, a feature value attribution model is created based on known false negatives made by the model. Shapley values, state-of-the-art values used for explaining model predictions, are employed to find out which features push the model towards making such errors. Lastly, we calculate one more score based on the input of 8 human experts in the field of fraud classification. Based on how important these experts deem certain features to be, we adjust the score per feature. The more important these experts deem a feature to be for their decisions towards a transaction being fraud, the more we penalize feature values differing from the norm. We call this the perceptibility score. We then calculate the total success score of a sample as the sum of all the aforementioned success scores per feature value.

As the last step of our pipeline, we take the top adversarial synthetic fraudulent samples and use these to augment our training dataset. In practice, this means that we add a number of synthetic samples to our existing training dataset. A supervised model trained on such a dataset is called *weakly-supervised*. The results show that this approach not only serves to improve the privacy preservation across the training dataset, but also significantly improves the degree in which the trained model is able to generalize to never-seen-before fraudulent transactions. The obtained results show that models trained on existing training dataset along with the top adversarial examples outperform their synthetic-sample-free counterparts by correctly classifying 7% more fraudulent transactions. This means that our approach shows a false negative rate of roughly 14% as compared to the default situation with a false negative rate of roughly 21%. This improvement does come at the cost of a 1% increase of false positives. To remedy this, we introduce 2 separate systems with the goal of reducing this number through external means. Finally, this work shows that in a real-world scenario, applying this pipeline leads to a 1/6 decrease in monetary loss incurred due to financial fraud.

This work aims to employ these methods and remedy the unique challenges in the field of financial fraud detection by answering the following research question and sub-questions:

1. **How can we leverage data-driven methods and human expert knowledge to improve the robustness of a supervised machine learning model to never-seen-before fraudulent transactions?**
    (a) How does one generate high-quality synthetic fraudulent transactions with a high machine learning utility and privacy guarantee?
    (b) How does one leverage both domain-knowledge and data-driven methods to obtain adversarial examples representing never-seen-before fraudulent behavior from such synthetic samples?
    (c) How do we use such adversarial examples to improve the generalizability and overall performance of an existing fraud detection model?
    (d) How does one leverage insights provided by an existing rule-based system for fraud detection to reduce the cost in terms of false positives incurred by the proposed method?
    (e) How does one integrate this method into an existing fraud detection architecture to test and evaluate it’s performance?

To answer these questions, this work will first provide the reader with an extensive background and related literature research divided into three chapters. The first chapter provides a detailed description of financial fraud and the fraud prevention measures Dutch financial institutions are required to take by the DNB. The second literature chapter gives an in-depth introduction to all machine learning techniques that this work relies on, consisting of both supervised and unsupervised machine learning approaches. The last literature chapter goes into detail on the implications on mistakes made by automated decision makers for financial fraud and dives deeper into the ethical side of the problem. The literature chapters are followed by an introduction to the high-level concepts associated with automated fraud detection in our case-study chapter. Afterwards, each part of the proposed pipeline is introduced in a logical order in multiple method chapters. Lastly, we conclude this thesis by showing the relevant performance metrics of each step of the pipeline in the experimental results chapter and discuss the implications of these results in the discussion chapter.

3 Literature Research: Financial Fraud

3.1. Financial fraud

The Oxford dictionary describes fraud as being ”the crime of cheating someone in order to get money or goods illegally”, which in itself is a very broad broad range of activities. The Section 326 of the Dutch Criminal Code (Wetboek van Strafrecht) defines fraud in the following way: “Any person who, with the intention of benefitting himself or another person unlawfully, either by assuming a false name or a false capacity, or by cunning manoeuvres, or by a tissue of lies, induces a person to hand over any property, to render a service, to make available data, to incur a debt or relinquish a claim, shall be guilty of fraud” [74]. Because fraud is not bound to single activity or statutory offence, it is difficult to devote an academic discipline to addressing the subject [29]. In this work we consider *financial fraud*: an instance of fraud where the list of possible violations is extensive and its consequences have a major impact on people from many corners of society. As certain types of fraud become more prevalent, tools are developed with the goal of addressing said fraud, which in turn leads to fraudsters developing new and harder to detect fraudulent methods. This enters us in an endless cat-and-mouse game where we, the cat, are often one step behind.

Much like the general term fraud, financial fraud is a broad term that envelops a multitude of criminal activities that all share the same goal: obtaining financial gain under a false pretense. The Dutch public prosecutor’s office categorizes financial fraud into two main categories [68]:

- **Horizontal fraud:** fraud that victimizes citizens and companies; and
- **Vertical fraud:** fraud that is committed using government money.

This is an important distinction to make, since the latter logically receives significantly more attention from the government. Even just in the year 2018 the Fiscale Inlichtingen- en Opsporingsdienst, the dutch Tax information and investigation service, spent roughly 1.060.000 hours combating fraud [68]. The majority of the fraud cases found by this organisation was associated with tax evasion. In our context however, we are considering financial fraud and transactions made by users of a financial service. Consequently, this work will only consider horizontal fraud.

Since 2015 horizontal fraud received increased priority by the dutch policy and because of that has seen a sharp increase in the number of suspects found [68]. Furthermore, the dutch public prosecutors office is investing more resources into fraud prevention as fraud becomes increasingly more organised, digitized, internationalized and complex. This trend is also shown in 3.1, where we can see a sharp increase over the past 10 years in 3 commonly occurring instances of financial fraud:

1. **Phishing**: Obtaining someone’s private information through lies and deception to, in turn, use this information for financial gain;
2. **Stolen bank cards**: Withdrawal of funds through stolen or lost bank cards; and
3. **Credit card fraud**: Fraud committed by using a credit or debit card, usually to transfer funds to accounts owned by criminals.

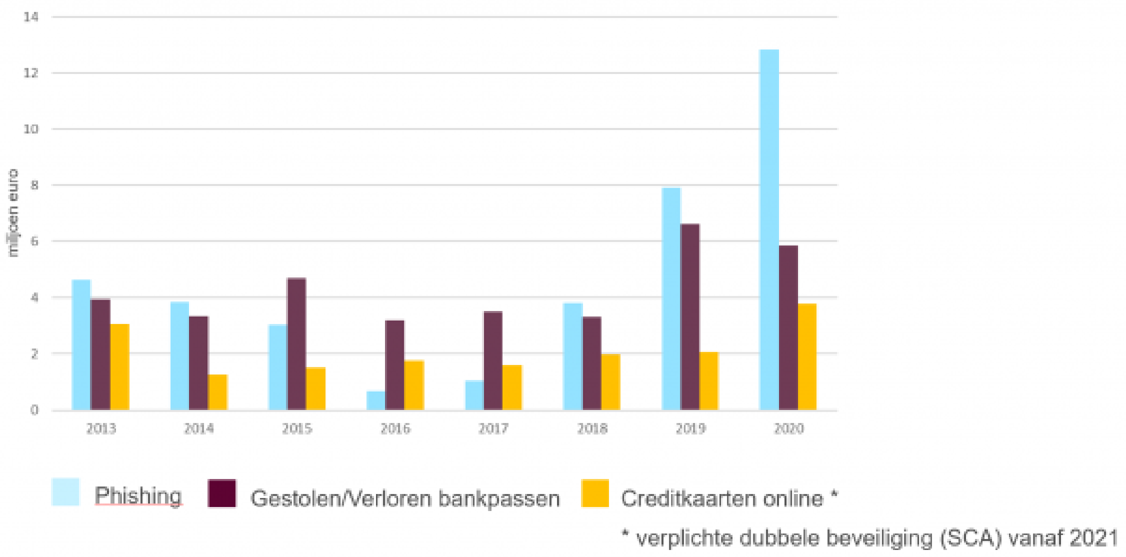


Figure 3.1: Damage caused by 3 types of fraudulent schemes over the past decade [7].

Financial fraud is of course not limited to these 3 examples. In the literature a more refined distinction between the existing types of financial fraud is given. These fraud types are defined as [83]:

- **False financial disclosures**: a variety of behaviors where the perpetrators make false statements about the financial health or performance of their financial institution or fund. The person or company involved does not necessarily have to be illegal, in fact many of the fraud committed of this type is perpetrated by otherwise legitimate parties;
- **Financial scams**: a scheme set up by a fraudster where a person is tricked into voluntarily handing over private sensitive information related to their personal finances or funds; and
- **Financial mis-selling**: practices that are intentionally misrepresenting or manipulating while selling or advertising a service or product to a customer with full knowledge that said product does not satisfy the end-users needs.

False financial disclosures and scams are similar in the fact that both methods are based on plain lies, but differ in the fact that scams are are hiding behind an enterprise that only seems legitimate, but is in fact illegitimate. In contrast, mis-selling use clever wording and other deceptive techniques to bend the facts in order to mislead people unfortunate enough to believe these mis-statements.

With the advent of electronic banking becoming increasingly popular, along with the rise of e-commerce, a multitude of new channels have opened for fraud of the financial scam type [93]. One such scam is financial identity scam, where victims are coaxed into handing over personal data, which is then used to make fraudulent money transfers [83]. This financially identifying data is often obtained through techniques such as spoofed e-mails directing a user to a counterfeit website with a prompt to enter their information or a person acting under the false pretense of being an employee of a bank that the user is a customer of. In some cases fraudulent schemes even make use of so-called *money mules*: people being payed or duped into receiving and forwarding fraudulent funds, most often through irrevocable payment services. Money mules are especially difficult to detect, since they are, aside from the current activity, by all means a legitimate user.

3.1.1. Impact of financial fraud

As the scale and prevalence of multiple types of fraud continues to increase, so does the impact on those duped by it. To more accurately be able to sketch a picture of this impact, we will divide the impact of fraud into two categories. First there is the impact on the person or user that is being duped. Second is the impact of fraud on the institution involved in the fraudulent transactions. For the former group to be victimized by fraud is especially heinous, as types of fraud such as scams can have serious implications for a persons financial well-being. In this context, financial well-being is defined by Drew et al. as ”a state of being financially healthy, happy, and free from worry” [31]. In the Netherlands, in the year 2020 alone close to 50 million euros of fraud traffic have been reported [68]. Compared to 2019 this is an enormous increase of **195%**. The majority of the committed fraud (roughly 60%) was of the second type, a financial scam in the form of phishing or so-called phone number spoofing, a fraud technique based on social engineering. A commonality between the vast majority of the aforementioned fraud instances is the fact that up to a certain degree, the responsibility lies with the financial institution to ensure that no fraud can be committed through their platform. A common view in the literature on financial scandals involving fraud is that they are symptomatic of deficiencies in corporate governance [37]. As a consequence, a widespread change in regulations and governance requirements across many countries has been set into motion over the past decade. In the Netherlands, the most recent of these changes is the *Guideline on the Anti-Money Laundering and Anti-Terrorist Financing Act and the Sanctions Act*. The next section will dive into more detail on this act.

3.2. Dutch financial institutions and the DNB

De Nederlandsche Bank (DNB), the Dutch national bank, supervises a wide range of financial institutions in the Netherlands. Through supervision based on a wide variety of acts, such as the *Financial Supervision Act* (Wft) and the *Anti-Money Laundering and Anti-Terrorist Financing Act* (Wwft), the DNB aims to uphold integrity in the services provided by these institutions. In this context integrity means, among other things, that the service is not used for money laundering, terrorist financing or other violations of sanction regulations. The DNB holds the responsibility of implementing and enforcing the Wwft [23]. Not only is integrity one of the pillars of trust for the proper functioning of financial institutions, integrity is also an essential part of running an ethical operation. The ethical implications of fraud and automated fraud detection systems will be discussed in more detail in chapter 5.

In order to uphold these standards of integrity, the financial institution first and foremost has to set up proper processes, procedures and measures in order to mitigate the risks of the aforementioned integrity-harming practices. Such measures include integrity policies, strategic reviews, mission statements and business principles. More concretely, in terms of fraud-prevention, the DNB expects a financial institution in the Netherlands to create codes of conduct for the customers and procedures to effectively control the various integrity risks. The regulatory framework that is deemed appropriate by the DNB for controlling money laundering and terrorist financing is *risk-based* [23]. As such, the precautionary means that financial institutions are expected to take are associated with the risk posed by the various customers. In turn, this risk is based on customer characteristics such as the country of residence, past transactions or sectors of profession. These characteristics are known as *risk factors*. The full extent of risk factors and how they relate to automated fraud detection will be discussed in great detail in multiple method chapters of this work.

To prevent the involvement in money laundering or terrorism funding, a financial institution is expected to identify any inherent identity risks resulting from the susceptibility of its services to be used for these means. To this end, institutions are obligated to draw up a *systematic analysis of the integrity risks* (SIRA). This assessment is comprised of four steps [23]:

1. **Risk identification**: identifying the risk factors associated with a specific customer, along with the areas of the provided service that are most susceptible to integrity risk;
2. **Risk analysis**: analyze the identified integrity risks to assess the associated impact and likelihood. Based on these, weights should be determined for the respective risk factor;
3. **Risk control**: designing a control procedure to deal with high risk cases or scenarios; and
4. **Risk monitoring and review**: monitoring and reviewing the identification and control procedure to improve the process or obtain further insights on fraudulent behavior.

The concrete content and interpretation of SIRA differs per specific financial institution. Larger institutions with a more complex set of services that are being offered that is also operating on international markets has to deal with more intricate risk factors than a small institution only operating in the Netherlands. The fourth step of SIRA, which comprises the main focus of this work, will now be discussed in more detail.

3.3. Transaction monitoring

Whereas the first 3 steps of SIRA are associated with customer compliance and the legal side of the fraud prevention spectrum, the fourth step envelops the literature and context we study the field of automated fraud detection. Financial institutions are required to take measures to prevent the flow of illegal activities through their platform. To this end, they must pay particular attention to unusual and suspicious transactions or patterns in transactions. If there are sufficient grounds for assuming that a certain transaction or string of transactions are associated with illicit activities, these transactions must be reported to the FIU. To effectively carry out such a monitoring process, larger financial institutions are required to make use of a transaction monitoring system and software. Such a system will generate alerts, which in turn will be investigated by human auditors trained in the field of fraud detection [23]. In other words, this system will flag transactions it deems suspicious and let ordinary transactions pass.

The Wwft states that such a system must include, at the very least, a set of predefined *business rules*. These rules must be in the form of scenarios with threshold values [23]. In more understandable terms, a business rule has to be intended to combat a certain instance of fraudulent behavior by checking a specific set of characteristics per transaction, based on which a score is calculated. If such a score exceeds a pre-defined threshold value, the transaction is flagged as potentially fraudulent. Chapter 6 will go into more detail on rule based systems, provide the reader with plenty of real-world examples to get acquainted with such a system and discuss the main shortcomings of a rule-based system.

Lastly, the Wwft states that financial institutions deem it necessary, they may make use of highly advanced systems on top of the rule based system. Usually such an advanced system becomes necessary as the financial institution grows and the nature of the average transaction becomes more intricate. When making use of such a system, the DNB requires the financial institution to possess sufficient knowledge to create, maintain and use the system. On top of this, the institution must be able to demonstrate its quality and effectiveness and is responsible for its performance. As such, this system may not be created by external suppliers [23]. This responsibility for the performance of the system, especially the classification mistakes made, creates the need for understanding the ethical implications of the usage of such an intricate automated fraud detection system. Chapter 5 is dedicated to discussing the ethical implications of fraud detection both in terms of classification mistakes and user privacy.

4 Literature Research: Machine Learning

The goal of this chapter is to provide the reader with a logically structured overview and introduction to the background literature that this work is based on. This chapter will first introduce the reader to supervised learning with decision trees, which are especially useful when working with tabular datasets and are a common occurrence when dealing with transaction-related data. Then, it will dive deeper into the interpretation and explanation of the results produced by such models and in particular one category of mistakes made by such models: *unknown unknowns*. Lastly, unsupervised learning techniques and their application for the generation of synthetic data are discussed. The chapter concludes with a link to the next chapter, where the privacy implications of data-mining using supervised methods, along with the effect of using synthetic data to augment the training dataset are discussed.

4.1. Decision tree classification

4.1.1. Decision trees

Decision trees have been around for many decades. From its inception, the decision tree has had tremendous potential both in terms of assisting decision-makers as well as explaining these decisions to parties less up-to-date with the specifics, for instance management [65]. Decision trees employ a top-down, divide-and-conquer strategy that partition each step of a decision making process into smaller subsets. In its most basic form, the decision tree is a rooted, directed tree consisting of two types of nodes: internal (or test) nodes and leaf (or decision) nodes [84]. In such a tree, the input is split along each internal node into two or more subsections of the tree based on some discrete function. Important to note is that each test node only considers one attribute of the decision making process in its test. Such decision trees are able to incorporate both categorical and numerical data. In case of categorical attributes the instance space is partitioned according to each of the attributes value. For numerical attributes the conditions are often based on a range. These partition methods will be discussed in more detail later in this section. Consider the following very simple example as an illustration: one has to host a cocktail party. The host needs to make the decision whether the party will be held inside or outside. Furthermore, the possibility of the weather being rainy exists. Each possible course along with its outcome and expected payoff is shown in figure 4.1. By modeling each distinct attribute of a decision making process, whether it is a decision or a chance event, such a tree is able to show a clear path towards an outcome while detailing choices, risks, information needs or monetary gains on the way.

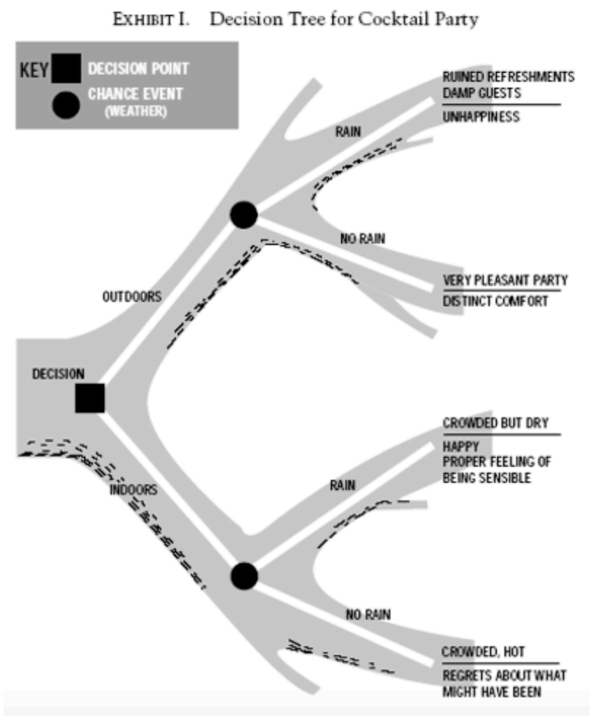


Figure 4.1: Decision tree for the cocktail party [65]

While decision trees were designed with the intent of helping decision makers with their work, they do not explicitly give the answer. Rather, decision trees provide the means to compare the consequences of different courses of action. J. Quinlan, who introduced decision trees to assist with decision making problems, describes the following advantages [81]:

- **Clarity and conciseness**: The entire classification process is easy to understand and scrutinize by a human decision maker;
- **Context sensitivity**: On their own, some attributes used by the decision tree may not be helpful in many contexts. Decision trees on the other hand allow for these attributes to obtain relevance many different contexts; and
- **Flexibility**: This method is able to adapt to multiple input data types, such as categorical or numerical data.

The first distinct advantage of decision trees will be explained in more detail in section 4.2 of this chapter. The second advantage comes as a result of the structural design of decision trees. A classification done by a decision tree can be traced back along one of the paths. Each testing node along this path is dependent on the outcomes of the tests before it. As a result, each testing node exists in its own context within the decision tree. These contexts provide different spaces in which many attributes can become relevant. As an illustration, we go back to our previous example. Imagine another decision point, where the organizer of the cocktail party has a certain budget to rent a venue and has to make a decision based on this number. This decision only becomes relevant when the decision is made to hold the party inside. If not for the way the decision tree is structured, this attribute would hold no value to the decision making process.

Many statistical classification methods often make use of either numerical or categorical attributes, which are continuous and discrete respectively. For instance Bayesian methods require all attributes to have a discrete value, meaning that numerical attributes need to be divided into sub-ranges or otherwise encoded [65]. Decision trees do not suffer from this type of inflexibility, as both discrete and continuous attributes can be handled gracefully. If there is the need to cope with n different discrete values, we can simply introduce n decision edges in our test node. For continuous numerical values the decision-maker can introduce different thresholds or ranges to determine the decision path in the test node.

Furthermore, Quinlan describes *restricted formalism*, the fact that each test node is limited to just one attribute, to be the main downside of decision trees [81]. The decision making quality of a decision tree heavily depends on a appropriate collection of attributes being provided. For instance, assume we have 2 continuous attributes X and Y. Assume in the decision making process a decision needs to be made based on whether X is greater than Y. Since test nodes are restricted to a single attribute, this forces us to include an, otherwise redundant, attribute relevant for this type of classification, in this example the attribute $X - Y$.

Induction of decision trees

The advantages described above show that decision trees provide powerful, comprehensible and flexible means for decision making and by extension automated classification. Continuing on the notion that the importance of providing an appropriate collection of attributes is crucial for constructing an accurate decision making process, we shift our focus from basic decision making to knowledge-based expert systems. In such a system, there exists a universe of *objects* that can be described in terms of *attributes*. Each attribute represents a discrete, mutually exclusive important feature of an object [79]. Looking back at our earlier example, this means the attributes *outdoors* and *no rain* are the values for the attributes *location* and *weather*, which describe the object *distinct comfort*. Furthermore, we can group each of these objects into mutually exclusive classes. In our example we could for instance introduce the classes: *party is a success* and *party is a failure*. Then, the objects *distinct comfort* and *happy, proper feeling of being sensible* are grouped in the former, while *unhappiness* and *regrets about what could have been* belong to the latter class. Such classes are often referred to as *positive instances* and *negative instances*.

With the definition of the universe as one ingredient, the second major ingredient is a proper *training set* consisting of objects of which the corresponding class is known [79]. Provided with these, we are now presented with an *induction* task: to construct a *classification rule* that is able to accurately determine the class of an object expressed as attribute values. For such a training set to be deemed adequate, it needs to satisfy one major constraint: if two objects share identical values for their attributes, they must belong to the same class as well. If this is not the case, it becomes impossible for the classifier to differentiate between these objects based on their attributes and as a result, the induction task becomes impossible as well.

As the size of the training set and thus the number of rules necessary for the decision tree to represent grows, the complexity of the of the decision tree grows with it [84]. We can express the size and complexity of a decision tree in terms of:

- The total number of nodes
- The total number of leaves
- The depth of the tree: the number of edges one needs to traverse to reach the leaf node from the root node
- The total number of attributes used

The problem of constructing a minimal decision tree that is able to represent the training set has been shown by Hancock et al. to be an NP-hard problem [48]. The authors further show that the theoretically best learning algorithm [35], is able to achieve a *Probably Approximately Correct* (PAC) decision tree of size n in $O(n^{log(n)})$ time and training set samples, n being the amount of attributes. The authors state their belief that smaller, less complex decision trees are a good representation of human comprehensibility. Furthermore, Breiman et al. show that the complexity of a decision tree has a crucial effect on it's classification accuracy [14]. One has to impose a limit on the complexity of a decision tree, to avoid overfitting on the objects in the training set [56] and in turn losing predictive accuracy on never-seen-before objects. These observations suggest that accurate, comprehensible approaches for constructing decision trees only seem feasible on smaller problems.

To construct a decision tree and, in turn, evaluate the quality of the found tree, numerous heuristics have been developed. These heuristic criteria cover a wide range of points-of-view from which to assess the quality of the decision tree. These viewpoints can be categorized into roughly 3 major groups [81]:

- **Information provided**: this group envelops heuristics most often based on the metrics *information gain* and *entropy*. Both of these metrics provide insight in how well a test node conveys information in terms of determining class membership of objects [82];

- **The degree of error**. Heuristics belonging to this group allow comparison in the errors made in terms of matching the input objects to the right class between test nodes. A common metric used is the *Gini Diversity Index* [89], which measures the expected error while taking relationships between the different classes into account; and

- **The statistical significance of a decision**. This type of heuristic compares test nodes in terms of its significance for determining the confidence with which a certain class can be excluded. Test nodes with a lower significance are deemed less desirable.

Choosing suitable heuristics for growing decision tree is of utmost importance in the construction of a decision tree, as subtle changes in these methods can significantly affect the trees produced [81]. Many characteristics of the training set play an important role in this process, such as the number of samples or the number of possible outcomes of a test node.

Reducing the complexity of decision trees

Attributes found ”in the wild” are often far from perfectly suitable for the construction of decision trees. The seemingly random nature of real-world samples lead to inadequacies in training sets, such as objects with unknown values for some attributes or the some values for attributes are erroneous or noisy [56]. In order to still efficiently leverage the advantages of decision tree classification methods on real-world tabular (containing both discrete and continuous attributes) data, methods to mitigate the growing complexity resulting from these inadequacies are needed. To this end, various methods, such as introducing *stopping criteria* and *pruning* have been developed. Stopping criteria prevent the tree from growing further based on a set of rules [67]. These rules can be based on features of the tree, for instance the depth of the tree or the amount of leaves, or statistical measures, such as not allowing an attribute to be tested further if the confidence of the attribute being independent of the object’s class is sufficiently low [81].

As opposed to stopping early, the act of pruning decision trees, as described by J. Quinlan, is ”to allow the tree to grow without constraint, then to remove unimportant or unsubstantiated portions [of the tree]” [81]. When pruning a sub-section of the tree, a quantitative method is applied to make the decision whether to replace the sub-tree with a leaf. Similar to the heuristics for growing the tree, various metrics exist as a base for pruning. Also similar to growing heuristics, these methods are based on the loss of accuracy (cost complexity pruning), error rates (pessimistic pruning) and loss of information associated with pruning a certain section of the tree. Pruning methods result in smaller trees, lower complexity and less overfitting resulting in better performance on unseen objects [80]. More techniques exist to make a tree more widely applicable and robust to never-seen-before objects. One such technique is *cross-validation*, which, when carefully integrated in the construction of the tree, reduces bias (and in turn overfitting) in exchange for a relatively low computational overhead [10].

This section has introduced the notion of decision trees, along with an overview of the advantages and challenges in this field. The next section will discuss a concrete application of decision trees for machine learning purposes, on which the contributions of this thesis heavily rely.

4.1.2. Tree ensembles

In order to efficiently leverage the advantages of decision tree classification methods on large, real-world tabular (containing both discrete and continuous attributes) datasets, a suitable application which leverages the aforementioned advantages and mitigates the shortcomings is needed. While single decision trees can be a very powerful tool for decision making and classification, even better accuracy can be obtained by using multiple decision trees in conjunction. In fact, ensembles of decision trees are among the highest performing types of classifiers [17]. Two widely applied techniques that make use of ensembles of decision trees are *random forests* and *boosting*. This section will briefly discuss the advantages and shortcomings both methods, while diving into more detail on the latter in support of the upcoming chapters of this work.

Random forests
As the name forest suggests, random forest classifiers use a collection of of multiple decision tree predictors, where each decision tree is fed a different set of attributes sampled independently and from the same distribution. More formally: a random forest consists of k tree-structured classifiers, where for each tree a random vector ω_k is generated. Based on this vector and the training set a tree is grown. For each classification, the trees cast a vote and the most popular class is chosen [13]. As a result, the effects of single trees overfitting to the training dataset are mitigated. By increasing the number k of trees used to construct the forest the generalization error, which describes how well the model can predict unseen data, converges to a limit.

One of the main advantages of random forests is how this method can handle categorical values with a high cardinality. Where, at the time, existing methods could find an optimal split for these values in $O(2^{N-1})$ time, N being the number of possible values, random forests simply handle this by selecting a random subset for this attribute for each tree [13]. Furthermore, because the trees are grown in a randomized fashion, there is no need for pruning, which can be a cumbersome task [52]. However, random forests are limited in the context of regression. For regression problems, the classification of a random forest is the result of the mean of the classifications of the k trees. Because of this, random forests tend to underestimate higher values, while overestimating lower values resulting in bias in its classifications [52].

Boosting
In the context of machine learning, boosting is a technique that aims to combine multiple weak classifiers into single strong classifiers. The notions of weak classifiers, strong classifiers and PAC learning form the basis for the concept of boosting [98]. In this framework, Valiant introduces a class to be *weakly-learnable* by a classifier if, given it has access to a source of examples (or training set), it is able to generate predictions that consistently slightly outperform random guessing. A class is said to be *strongly-learnable* by the classifier if it can output a prediction that is correct on all but an arbitrarily low number of samples with a high probability. In his work, Shapire proved the *hypothesis boosting problem*, where he showed that these two notions of learnability are in fact equivalent, by creating a method that boosts the accuracy of a weak learner to that of a strong learner [86]. In contrast to the approach used by random forests, the natural approach of boosting the accuracy of weak learners by running the procedure multiple times and taking the majority vote or mean, was proven inadequate [55]. This reasoning set into motion the creation of the first provable, polynomial time boosting algorithms.

One of these algorithms is of particular interest, as it solved many of the practical difficulties that the early boosting algorithms suffered from, is called *Adaboost* [39]. The Adaboost algorithm takes a training set $(x_1, y_1), ..., (x_m, y_m)$ as an input where x_i represent attributes from the domain X and y_i a label from the set of classes Y. As this work focuses on binary classification, this section of the literature research will also only focus on that. As such, we assume that $Y = \{-1, +1\}$. To train on the given training set, Adaboost calls multiple rounds $t = 1, ..., T$ of a *weak-learning algorithm*, an example of such an algorithm consisting of tree growing and pruning steps discussed in the previous section. Between each round t, Adaboost maintains a weight $D_t(i)$ for each sample i of the training set. The weights are initiated equally. After each round, Adaboost increases the weights of miss-classified samples, while decreasing the weight of correctly classified samples. As a result, the weak-learning algorithm in future rounds is forced to focus more on these samples. For each of the weight distributions used per round, the weak-learning algorithm finds a *weak-hypothesis* $h_t : X \longrightarrow \{-1, +1\}$, the quality of which is measured by its *error*, which is defined as:

$$\epsilon_t = Pr_{i\ D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$$

As this error gets smaller, Adaboost assigns an importance measure α_t to each weak hypothesis, where $\alpha \geq 0$ if $\epsilon_t \leq 1/2$ and α_t increases as ϵ_t decreases. Finally, Adaboost finds a *final hypothesis* H as the weighted majority vote of every round T, with α_t functioning as the weights.

Using this method of boosting has many advantages: it is simple to understand, has little parameters to tune and requires very little to no prior knowledge about the weak-learner that is used [39]. Furthermore, given it is provided with a weak-learner that can reliably create weak-hypotheses and a training set of sufficient size, this method is guaranteed to improve accuracy and reduce overfitting. On the other hand, as a result of the necessity of a proper training set and reliable weak-learner, the performance of the boosting method is heavily reliant on these factors. On top of that, boosting is especially vulnerable to noise [27]. The first of these caveats is of special interest, as this work introduces a novel method that aims increase the quality of training sets that are supplied to a boosting approach for classification. The rest of this section will go into more detail on a state-of-the-art approach to boosting for classification.

4.1.3. Gradient Boosting

While posing multiple advantages over single decision trees and other ensemble methods, one main caveat of the Adaboost approach to boosting is that it is purely algorithm-driven and as a result, making a detailed analysis of the performance and its properties an arduous task [87]. To be able to further analyze and increase the performance of such models, a connection to statistical frameworks is needed. To achieve this, a gradient-descent formulation of such models was created which were called *gradient boosting machines* (GBMs) [40]. As opposed to previous methods of boosting, in a gradient boosting approach, each round t aims to construct a new weak-learner that is maximally correlated with the negative gradient of the loss function ϵ_t [70]. This loss function can be arbitrarily chosen, providing the system designer with the freedom of modeling the learner for his specific domain or problem. This freedom and flexibility that GBMs provide make them highly customizable for a myriad of data-driven tasks. As a result, over the last decade GBMs have shown considerable success both in the field of data-mining and practical applications of machine learning [78] [63] [97].

Lastly, as this section has slowly entered the state-of-the-art field of decision tree applications for machine learning, one more boosting technique will be discussed. As applications of gradient boosting machines move into the open world, many new challenges arise such as incomplete or noisy data, unmanageably large datasets and the every changing, seemingly almost random, nature of prediction classes. In order to construct a predictor that can perform in these ”in the wild” scenarios, one requires a scaleable construction method that produces generally applicable models. One such method is *Regularized Gradient Boosting* or *eXtreme Gradient Boosting* (XGB) [19]. The authors describe the main advantage of their approach as ”its scalability in all scenarios”, as the system runs more than 10 times faster than existing similar solutions. On top of this, they describe their approach as being optimzed for *sparse datasets*. By combining these advantages, XGB is can be applied to solve real-world problems, while using a minimal amount of resources. Chapters 7 and **??** will go into detail on how XGB models are used and implemented in this work. The next section of this literature chapter will take a deep dive into how we interpret predictions and errors made by such classifiers in this work from a high level point of view.

4.2. Interpreting and explaining classification results

In the year leading up to the US invasion of Iraq the US Secretary of Defense, Donald Rumsfeld, made a statement that caused quite some commotion. When asked about the evidence surrounding the US involvement in the supply of weapons to terrorist groups, he stated:

”Reports that say that something hasn’t happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don’t know we don’t know. And if one looks throughout the history of our country and other free countries, it is the latter category that tend to be the difficult ones.” [73]

The vagueness of this statement led to widespread ridicule at the time. Even though in the context of the question asked this might have been justified, from a more general point of view his statement does make sense. There ***are*** things we don’t know we don’t know and as a result, are very difficult to identify. While the former secretary had introduced *unknown unknowns* (UUs) to a considerable new audience, the notion of unknown unknowns was nothing new.

The concept of *known knowns*, unknown unknowns and every other combination originates from the *Johari Window*, a graphic model created by Joseph Luft and Harry Ingham in an attempt to visualize the difficulties in thinking about one’s own behavior in relation to others [64]. This window, consisting of 4 quadrants, provides us with a framework to critically think about the awareness of knowledge at one’s disposal. The top left quadrant, the *area of free activity* (known knowns), refers to the information known by oneself and others. The top right quadrant, the *blind area* (unknown knowns), is the information known to others but not to ourselves. To the opposite end of the window is the *hidden area* (known unknowns), the things we know but is unknown to others. Lastly, and most interestingly, there is the *Area of unknown activity* (unknown unknowns). The information described by this quadrant is not known to anyone, yet we can assume this information exists, as over time the information from this quadrant becomes known. As this information gradually comes to light, it is key to recognize that this newly acquired information and its influence had been there all along.

As predictive models based on *machine learning* are in essence a computerized version of the human way of learning, we find that these definitions are an effective way of categorizing the knowledge that a model holds. When presented with a certain string of inputs, a trained model will output a prediction with a corresponding confidence score. Based on this information we are able to translate these outputs to any of the 4 quadrants described earlier. Using the confidence score attached to a prediction, usually a number between 0 and 1, along with predetermined thresholds to indicate the confidence level, we can decide whether the model is confident in its prediction or not. A human expert can then determine if the classification is correct. One of the root causes for the existence of UUs in supervised machine learning is the mismatch between the data used for training the model and the data found after during deployment of the model [60]. More specifically, the existence of UUs can be a result of systemic bias in the training data. One such cause is *underrepresentation* or complete absence of certain instances in the training data. As an example of this, imagine an image-based classifier to predict whether a picture depicts a dog or a cat. The training data consists of a collection of black dogs and white cats. After training on this dataset, if the model is presented with an image of a white dog, it will likely incorrectly label it as a cat with a high confidence score [60]. Even though there are many distinctive features between dogs and cats, such as shape of the nose or ears, as a result of this bias the model might learn to make predictions exclusively based on fur color. As a result of under-representation in the training data, pictures of white dogs are UUs to the model. Not only under-representation can be a cause of this bias, but also a shift in population can lead to old training data no longer being representative for data found today [20]. To illustrate, again consider the model used to classify transactions. As fraudsters think of new techniques to stay ahead of the curve all the time, the data used to train this model needs to be representative of this. Knowledge of fraudulent payments from 10 years ago is not likely to help detect fraud based on current day strategies.

This section aims to provide the reader with a detailed overview of the current state-of-the-art surrounding the detection of UUs in the context of machine learning, provide insights on how the newfound knowledge about these UUs can help improve the quality of machine learning applications and create an high-level comparison between existing state-of-the-art techniques. Lastly, it links the notion of UUs in the context of fraud detection to the contributions made in this work.

4.2.1. Unknown unknowns and model robustness

Nowadays, a significant amount of predictions, decisions and estimations for business and government organisations are made by machine learning models. These models are employed in a wide variety of fields ranging from banks automatically checking if a payment that is made is fraudulent to making medical predictions to self-driving cars. On top of this, popular web service providers such as Amazon or Microsoft, have steadily been introducing pre-trained machine learning models over the past years. With the advent of such a significant amount of machine learning applications, it is of key importance to understand what is happening when such models make mistakes. No model, no matter how advanced, is perfect and is bound to make mistakes

The errors produced by a predictive ML model can be divided into two categories: known unknowns and unknown unknowns [1]. The former category has been studied thoroughly and a plethora of heuristics exist to address these types of errors. The common strategy for this is largely based on the same concept: select the predictions with the lowest confidence scores and use human annotators to confidently assign a ground truth label, also known as *active learning*. Examples of such methods are:

- **Uncertainty sampling:** Find instances for which the model has high uncertainty and and use human feedback to greatly reduce the amount of low confidence instances [62].
- **query-by-committee:** A group of people is trained on the same dataset. Then, the instance that results in the highest disagreement in the group, for instance a case where half the group answers ”yes” and the other half ”no”, is chosen. In turn, this instance can be labeled by an expert [90].
- **Near-separating hyperplane sampling:** Use a *support vector machine* (SVM) to divide the label space into two areas separated by a hyperplane functioning as the decision boundary. The points closest to this boundary correspond to the points with the highest uncertainty and are selected for training [88].

Even though these examples are helpful in increasing our understanding of the model and its weaknesses, they are of no use in finding the latter type of errors. In fact, by applying these methods we increase the amount of cases that the model is certain about, but in turn decrease the chance of finding UUs [1]. While known unknowns result from shortcomings of the model, UUs are a result of what T. Dietterich describes as ”unmodeled aspects of the world” [28]. He further goes on to explain that for a machine learning model to be truly robust, it has to conform to 5 standards of robustness:

1. Robustness to errors committed by human operators. For instance the scenario where a human and a ML model work together in a self-driving car.
2. Robustness to miss-specified goals, another type of human-caused error where a command to a ML model is misinterpreted for instance due to being vague.
3. Robustness to cyberattacks. As ML models become integrated into all kinds of services, such as financial markets or power grids, they must be resistant to various attacks on these systems.
4. Robustness to errors in the model itself. This standard comes down to becoming robust to known unknowns.
5. Robustness to the unmodeled aspects of the world.

This last form of robustness is of particular interest, as even if all other forms of robustness are addressed, the model will still continuously be confronted with this last type of problem, even more so in the field of financial fraud. Even though there is little information known beforehand about UUs in the wild, it has been found that high-confidence mistakes often follow a systematic pattern. In other words, many UUs have been found to be part of corresponding regions in the feature space of the classifier, as if they are the ”blind spots” of the model [3]. These superficial patterns are also referred to as *bias*. Models that suffer from a high bias seem to perform well on in-domain data, but struggle to perform when deployed in real situations and are easy to fool [21]. An example of this are models that find relations in text. These models often base their answers solely on the existence of certain keywords irrespective of the context as a result of bias [11]. Over the past years, progress has been made towards using knowledge about UUs to improve the overall quality and understanding of ML systems. The next sections, acting as almost a roadmap of this progress, provide an overview of the research done from identification of UUs to using this knowledge to filter bias from widely used ML datasets.

4.2.2. Identification of unknown unknowns

Human centered

One of the first and most prominent research done towards identifying UUs is the work done by Attenberg et al. [1]. The authors describe a technique for identifying these errors by making use of a game-like setting to leverage the natural skill of humans in detecting these kinds of errors. In their system, crowdworkers were asked to identify cases where they thought the model would make a classification error in exchange for a monetary reward for every correctly identified error. The authors found that even untrained humans were able to discover a large amount of cases where the model is confident about its prediction, but is incorrect. They conclude by mentioning that a logical next step is to use this knowledge of the vulnerabilities of the model to improve automatic decision making.

Human-in-the-loop

When talking about a human-in-the-loop system, it means that at some point in the ML pipeline there is human intervention to provide valuable information to the otherwise automatic process. One such system used to understand ML system behavior and identify failures is called *Pandora* [72]. In this system, content features and relationships between these features are clustered, either automatically or using human annotators. Then, based on these features and the model output, Pandora produces multiple views to characterize the circumstances that are most likely to lead to system failures. By doing so, this system aims to improve the accountability of ML systems. This is a problem that is becoming increasingly important as more and more ML applications are rolled out in the open world.

Lakkaraju et al. [60] take identifying ML failures one step further and propose the (to their knowledge) first algorithmic approach to the discovery of UUs. They remark that these types of errors often result from bias in the training data. As a solution, the authors propose a two-phase framework where the data is first partitioned based on model confidence and feature similarity, after which an explore-exploit approach is used to discover the UUs in these groups. When a potential UU is found, a human oracle is queried to find out the true label of the target. This approach focuses purely on the in- and output of the model and training dataset, meaning the ML model itself is treated as a black box.

As an extension to this research, Bansal & Weld argue that the above research yields an effective algorithm for discovery of UUs, but fails to provide a complete understanding of the limitations of the classifier in question [3]. The authors aim to improve on 3 weaknesses that they have found in the aforementioned research. First, their approach assigns more utility to UUs that are discovered and are distinguishable from previously discovered UUs. Secondly, regions of discovered UUs that represent a larger extent of blind spots are assigned a higher utility score. Lastly, the approach of Lakkarju et al. assigns the same utility to a very rare UU as it does to a UU from a very dense region of the feature space. As this dense region indicates a significant systemic collection of errors in the classifiers predictions, this extension prefers finding UUs in these areas. All things combined, their algorithm provides a better coverage of the regions in which UUs are found, thus providing a more broad overview of the blind spots of the model.

Algorithmic

While humans are naturally strong at recognizing and identifying UUs, relying on human input is costly. As a consequence, the utility functions of the approaches mentioned above all try to minimize the amount of queries to the oracle. The research done by Chung et al. assumes no existence of such an oracle [20]. In fact, it does not even assume the existence of labeled testing data at all. Their approach takes identification to the next step and introduces an approach to not only detect these errors, but aims to remedy them by leveraging the knowledge that these errors often result from discrepancy between the training and testing data (or, a more general term used by the authors: *covariate shift*). The authors describe a system where species-estimation techniques in conjunction with data-driven methods are used to estimate the feature values for the UUs of the model. Their results show that their proposed technique, in the presence of covariate shift, dramatically helps to improve model performance. Like the other approaches mentioned, the machine learning model is treated as a black box. The authors mention that an interesting direction for future work in this area would be towards how the properties of different ML models relate to the existence of UUs.

4.2.3. Addressing unknown unknowns

While identifying system failures and more specifically UUs help us understand the vulnerabilities of a system, the end-goal is to use this information to improve the existing ML models. As stated before, the existence of UUs originates from the discrepancies between the data used to train and to test, which we call *bias*. This phenomenon significantly inflates the performance of a model on training data, which leads to an overestimation of the actual capabilities of the model [61]. We want to try to find a way to leverage our knowledge about the UUs present in a system to reduce this bias as much as possible. As a result, we end up with a model that is more robust, generalized and ready for deployment ”in the wild”.

To this end, Clark et al. show that such knowledge about biases can be used to increase the robustness of a ML model in the case of such a domain shift [21]. Their approach consists of two stages: first, the authors train a model that makes predictions purely based on biases in a dataset. Second, another model is trained in ensemble with the first model, which encourages the second model to learn a different strategy. Then, only the second model is used on the test set. As can be expected, the first model performs very well on test data, but struggles on new, unseen instances. The authors results show that their approach outperforms a baseline model trained without any modifications in multiple test scenarios, ranging from synthetic data to the VQA dataset (a dataset containing open-ended questions about images).

A similar research done by Kaushik et al. focuses its efforts on so called *spurious patterns* [54]. These are underlying patterns and correlations in datasets that result in unintended relations between the input and output of a model. An example in the field of computer vision is the work done by Jo & Bengio where it is shown that deep neural networks unintentionally learn to use background clues or surface-level textures to recognize the subject of the picture [53]. To put this more concretely, the authors remark: *”the beach is not what makes a seagull a seagull”*. To address this, the authors propose a human-in-the-loop solution to manipulate the datasets with information that they call *counterfactual revision*. In practice, crowdworkers are asked to not to label instances, but instead to edit them to enrich the dataset with a counterfactual instance. To clarify, the authors apply this on a NLP model for sentiment analysis. The worker is directed to for instance revise a negative movie review and change it to a positive one, while changing as little information as possible. In doing so, the authors manage create enriched datasets to train more robust classifiers which are less reliant on spurious signals.

Lastly, a very recent and promising approach called *AFLite* (a lightweight framework for *adversarial filtering*) has been shown to effectively reduce bias in both synthetic and real datasets [61]. In their work, the authors recognize that most previous studies follow a *top-down* structure: the algorithms that reduce bias are mostly guided by domain-specific knowledge and intuition of the researchers. As a result, these algorithms are bound to the biases that are manually enumerated. To contrast these approaches, AFLite introduces a *bottom-up* approach: a model-based approach where the goal is to go beyond the human intuition and remove the spurious artifacts that a model, unbeknownst to the operators, uses. While the authors prove that this approach effectively reduces bias, improves model generalization and better ”in the wild” performance of models trained on the filtered data, AFLite does bring some new challenges to the table. Using AFLite makes applying widely used performance metrics and benchmarks considerably more challenging. For example: the training accuracy observed on one of the test datasets dropped from 93% to 63%, indicating that the original accuracy might have been inflated as compared to the ”in the wild” performance as a result of dataset bias.

When considering dataset filtering, the trade-off considerations shift from complexity versus generalization to the changes in performance due to the increased absence of bias. As briefly discussed earlier, filtering bias from a dataset prior to training leads to an increase in performance in the out-of-domain setting but comes at the cost of losing in-domain performance [21]. This is not surprising, as bias in datasets by definition increases the in-domain performance (which is the root cause of the problem in the first place). As a result of this, it becomes a laborious task to effectively compare performance to existing benchmarks. Training a model on a dataset filtered by adversarial filtering will very likely result in lower scores for common metrics, such as accuracy, compared to the same model trained on an unfiltered dataset. This means that for an equal comparison, all widely used AI benchmarks would need to be adjusted to include results based on filtered datasets. Many of the state-of-the-art research papers therefore suggest future works that cater to this need, such as: constructing suitable development datasets [21], releasing filtered dataset variants to further progress on benchmarks [61] or creating entirely new complex benchmarks [32]. One such approach is developed in this work, where the aim is to create a method to ”reveal” such blind spots with synthetic samples representing the UUs of the model to in turn improve generalizability.

4.2.4. Explaining model predictions

As the field of AI and automated decision making progresses, the need for justification of these decisions increases with it. No longer is it sufficient to just reach a correct decision based on an inscrutable black box system and human-decision makers needed to both be able to understand and explain how such a decision is reached [36]. As an example, for a medical application, a practitioner would prefer a decision-making tool that is correct 85% of the time, but being able to relate back to his own medical knowledge over a black box type tool that is correct 90% of the time [81]. For decision trees, this comes as a natural property of the decision making process. Rather than for instance statistical classification methods, where a classification is represented by a collection of numbers, decision trees represent classifications as a conjunction of decisions. It is not far-fetched that the latter is better explainable. This section will dive deeper into the explainability of the decisions made by tree-based methods, describing not only how to explain **how** a certain decision is reached, but also **why**.

The field of financial fraud detection is no exception from this increased need for explainability. As mentioned in the previous chapter, decisions made by an automated decision making system potentially have a huge impact on the financial well-being of a user. If someone is flagged as a fraudster, the financial institution is expected to be able to explain to the FIU how this decision was made [23]. For simple rule based systems this is fairly straightforward. As an example, consider the following simplified, fictitious transaction monitoring rule:

Flag a transaction made by a user as fraud if:

- The transaction was made from a high risk country **and**;
- The amount of the transaction is over 5000 euro **and**;
- The transaction was made to an organisation considered to be of high risk.

If a user gets flagged by this rule, whether he is a fraudster or not, the explanation of **why** the user has gotten flagged is trivial. Apparently this user has made a highly suspicious transaction and thus the decision can be justified. For machine learning models on the other hand, this decision is much more difficult to justify. A feature array based on the transaction goes in and a decision comes out. One cannot justify this decision by simply stating ”because the model said so”. For this reason, the need exists to be able to explain any decision made by the model in a human-understandable manner. This section will consider 2 state-of-the-art methods to formulate such explanations, which the contributions made by this work also build upon.

Shapley values

The *Shapley value* was first introduced by Lloyd S. Shapley as a way of calculating and describing the extent of the contributions of every player in an *n-person cooperative game* with regards to the final outcome of the game [91]. In a *non-cooperative game*, the game is defined as a sequence of moves each player can take leading to an unique possible play with its own associated payoff. As opposed to these games, cooperative games reduce this entire collection of data to a coalitional form [101]. The payoff values are no longer defined as on an individual plays, single actions or solitary moves, but rather on a single value that described the outcome of the game. Shapley introduced a single-point solution to associate a single payoff vector to a coalitional game which, at the time, were considered to be a black box due to the high complexity of such games.

More formally, a coalitional game played by a finite set of players $N = 1, 2, ..., n$ is a function v that maps the set S of all 2^N possible coalitions to a real number representing the total payoff. For a coalition consisting of no players, so $v(\emptyset)$, this value is 0. Furthermore, there exists an operator ϕ that assigns a vector of payoffs to each game defined as $\phi(v) = (\phi_1, \phi_2, ..., \phi_n)$. Each payoff $\phi_i(v)$ represents the payoff for player i in the game, which can alternatively be interpreted as the measure of player i's power (or influence) in the game. Shapley has defined the Shapley value as the average marginal contribution of player i to the game v as [101]:

$$\phi_i(v) = \frac{1}{n!}\sum_{\pi\in\Pi} v(p_\pi^i \cup i) - v(p_\pi^i)$$

In this equation, $p_\pi^i \cup i$ represents the set of players from N that precede player i in the game. Here, π represents a permutation on the set of players. This permutation can be interpreted as the order in which the players precede player i and is a one-to-one function on the set N. The set of all possible permutations is denoted by Π. Lastly, assuming that permutations are randomly and uniformly drawn from this set, then we have to include $\frac{1}{n!}$ in the function, since the set Π consists of $n!$ possible permutations. The most interesting part about the Shapley value is that this definition of the Shapley value represents an unique value that satisfies four axioms:

- **Efficiency**: $\sum_{i\in N}\phi_i(v) = v(N)$, the sum of all individual player i's payoffs is the total payoff of the game. In other words, each player in the coalition properly distributes itself among the total resources available in the game;

- **Symmetry**: in the game, if players i and j are symmetric, then $\phi_i(v) = \phi_j(v)$. Players i and j are symmetric if for **any** coalition S they make the same marginal contribution;

- **Dummy**: If a player i's marginal contribution to **any** coalition S is null, then player i is considered a dummy player and in turn $\phi_i(v) = 0$; and

- **Additivity**: For games v and w, the following additive relationship holds: $\phi(v+w) = \phi(v)+\phi(w)$. Here, $\phi(v + w)$ is defined by $(v + w)(S) = v(S) + w(S)$ for **any** coalition S.

This strong definition, the game-theoretic context and desirable properties of the Shapley value have lead to the Shapley value being adopted in the field of machine learning explainability. To conclude this section, an overview is given of **how** Shapley values can be interpreted as a stepping stone to one of the main contributions of this work.

Interpreting Shapley values

To interpret the Shapley values associated with a single prediction, we can formulate the following definition: The contribution ϕ_i of feature i to the prediction of a single sample compared to the average prediction of the complete dataset is expressed as the Shapley value [69]. Consider the following example as an illustration: we are training a random forest regression model to predict the number of bicycles rented on a specific day [69]. The tabular dataset contains both numerical and categorical features such as: the season, the weather situation and temperature. The count of the rented bicycles is the target class for this regression task. For a particular day, the explanations expressed in Shapley values could look like 4.2:

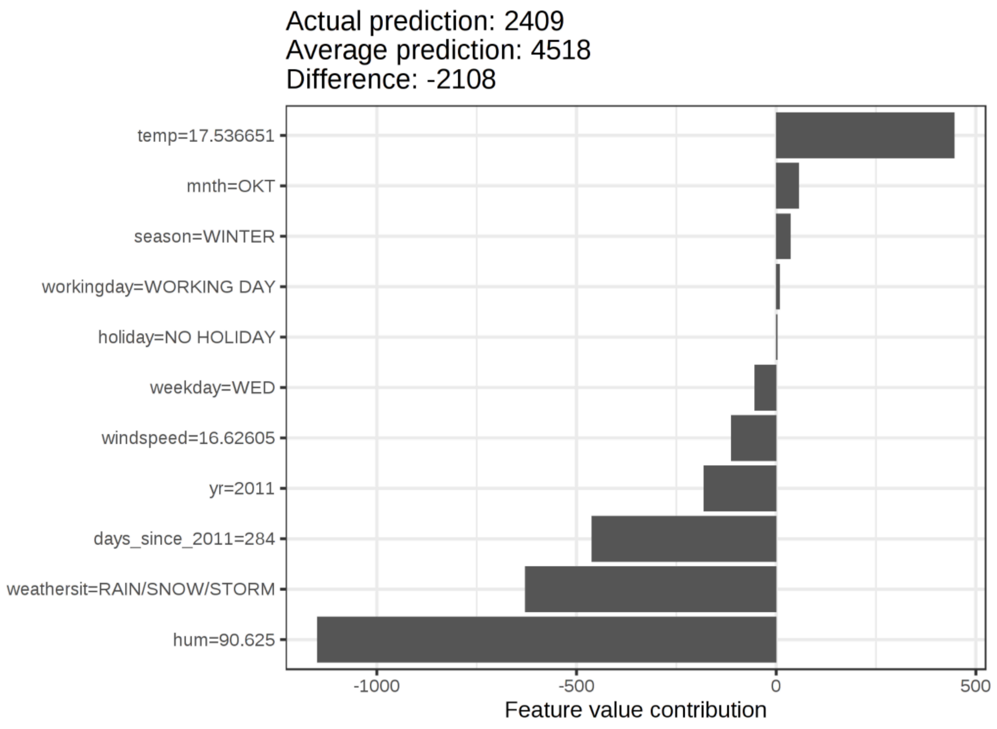


Figure 4.2: Shapley values for a particular day [69]

As we can see from figure 4.2, the average prediction for the number of bikes rented is 4518 and the actual prediction for this specific day is 2409. This difference of -2108 is expressed as feature value contributions in the bar chart. If we consider the most impactful feature contributions, the humidity and the temperature, it stands out that a high humidity has a negative impact on the amount of rented bikes, while a moderate temperature has a high positive impact on the amount of rented bikes. As it stands, Shapley values in essence *show the model designer the degree in which certain feature values push the model towards making a certain decision*. In terms of explainability, this is the main takeaway for this work and this way of thinking will be exploited in method chapter 9 to create adversarial examples in the context of fraudulent transactions.

Global Shapley values

So far, Shapley values in the context of a single prediction have been discussed. In this work, we are mostly interested in the effect of feature values over a large number of predictions. Where Shapley values are able to provide *local explainability* for one data point x, *Global Shapley values* are able to provide insights on the behavior of the model over the entire dataset [41]. Global Shapley values are calculated by taking the sum of the Shapley value for each individual data point in the dataset. Global Shapley values provide the model designer with insights on which features generally contribute to the decisions made by the model and in what way they influence the average model prediction. In method chapter 9 of this work, it is explained how global Shapley values are used and interpreted to gain interesting insights in the behavior of a fraud detection model.

4.3. Unsupervised learning for synthetic data generation

As was shown in section 4.2 of this chapter, supervised learning has a great range of applications in a wide variety of fields. As also mentioned in that section, one of the main challenges of supervised learning is the dependence on a properly labeled training dataset. Constructing a labeled dataset is often an arduous task, requiring both domain-specific knowledge and a high number of man-hours in the form of manual labeling. Even in the possession of such a labeled training dataset, one problem that still remains is the problem of class imbalance. Supervised machine learning approaches are known to suffer from this class imbalance, as many frameworks tend to favour the majority class. Automated classification systems in the field of financial fraud detection are no different and, as fraudulent transactions are significantly more rare than their legitimate counterparts, the challenge of working with a heavily imbalanced training dataset is even more prevalent. *Unsupervised learning*, as opposed to supervised learning, does not make use of such labels. Unlike other techniques that do not make use of class labels, such as *reinforcement learning* where one uses a reward function to point the learning process in the right direction, unsupervised learning only receives raw feature arrays as inputs [43]. While the idea of machine learning without target class labels might seem counter-intuitive, there are plenty of practical applications of unsupervised learning. In essence, one can think of unsupervised learning as a means of finding patterns in unstructured or seemingly noisy datasets. Examples of such applications are *clustering* and *dimensionality reduction* [4].

The majority of the work done in the field of unsupervised learning focuses on constructing a *probabilistic model* of the given dataset. More formally, consider a training dataset consisting of $n - 1$ observations $x_1, ..., x_{n-1}$. In many cases, it is useful to be able to estimate the probability distribution of a new entry x_n given the training dataset. The model learns $P(x_n|x_1, ..., x_{n-1})$, assuming that $x_1, ..., x_{n-1}$ are drawn from some joint distribution $Pr(x)$ in an identical and independent fashion. In turn, such a model can be used for *outlier detection*, *classification* or *data compression*, among many other purposes [4]. Another application of these probabilistic models that has been receiving a significant amount of attention in the recent years is the generation of *synthetic data* that has similar properties to the data found in the training dataset. In their work, Frasch et al. [38] describe a method for constructing a generator for training data used by supervised and unsupervised learning methods. They argue that using synthetic data for training purposes has two distinct advantages:

1. Collecting and/or labeling real-world data is in some cases very difficult or even impossible; and
2. Real-world variables found in real-world data are very hard to control, both in the problem-domain and in the feature-domain.

Even though their approach has many advantages, the main caveat is the fact that one needs full knowledge of the statistical characteristics of the problem at hand. Another similar approach, this time in the context of grammatical error detection, was proposed by Grundkiewicz et al. [46]. In their work, the authors enlarge the training dataset by generating additional synthetic sentences and generating synthetic sentences containing noise. First, their supervised model was pre-trained on the synthetic dataset, after which it was fine-tuned on an authentic, real-world dataset. This method has shown considerable success over similar methods, obtaining high rankings on multiple shared tasks in the field of grammatical error correction. Lastly, in a more recent research, Devaranjan et al. [26] propose a supervised method that leverages both synthetic and real-world target images to train an object detector. Furthermore, they use an unsupervised approach to generate the synthetic images and show that, without any supervision, their generator is able to produce data that captures key statistic structures much akin to real-world images. Their experiments on both real and toy datasets show a significant improvement of their model trained on the combined dataset compared to baseline datasets.

In this work, a state-of-the-art framework for creating synthetic data that has gained a lot of traction recently will be discussed and explained in detail. Then, an adaptation of this framework which allows us to generate realistic synthetic samples on tabular data, which poses additional challenges, will be described.

4.3.1. Generative Adversarial Networks

For a long time, the most promising work in the field of unsupervised learning has generally involved *deep-discriminative* learning models, where a high-dimensional input is mapped to a class label [59].

On the other hand, *deep-generative* models did not see the same striking successes, as there are two main difficulties that hinder efficient computations [45]:

- For strategies such as *maximum likelihood estimation*, the method of estimating the parameters of a probability distribution by maximizing a certain likelihood function, it is often very difficult to approximate the many intractable probabilistic computations.
- As opposed to the discriminative context, in the generative context it is difficult to utilize the advantages of piecewise linear units that are often seen in backpropagation or dropout algorithms.

To bypass these difficulties, Goodfellow et al. [45] have proposed a procedure for generative model estimation where a generative model is pitted against an adversary. *Generative Adversarial Networks* (GANs) employ both a discriminative and a generative model, where the goal of the generative model is to "beat" the discriminative model. More specifically, the discriminative model aims to learn whether a certain sample is generated by the generator, the model distribution, or part of the data distribution. One can regard the generative model as a band of counterfeiters, whose aim it is to produce as realistic counterfeit money as possible. In the same context, one can regard the discriminative model as the police, whose aim it is to distinguish between real and fake money. This cat-and-mouse game is driven by competition, as both teams will continue to increase their performance based to keep up with the other team. This section aims to give a detailed overview of how GANs work and assumes the reader has some basic prior knowledge of neural networks.

Formal description

In the most straightforward application the adversarial network consists of a *discriminator* and a *generator*, which are both multilayer perceptrons. The network takes a dataset x as input. The generator G is defined as a multilayer perceptron, which represents a differentiable function with parameters ω_G. This function is defined as the mapping $G(z; \omega_G)$ to the dataset space, where $p_z(z)$ are defined as the prior on input noise variables. This function allows for G to learn a distribution p_G over dataset x [45].

The discriminator is defined in comparable fashion, as a multilayer perceptron $D(x; \omega_D)$. As opposed to G, D only outputs a single scalar: the probability that a sample comes from the dataset x rather than p_G. D is trained to maximize the probability of assigning the correct label to samples fed to it from either x or p_G. G is trained simultaneously to D, but as opposed to D, G is trained to minimize $log(1-D(G(z)))$, which in essence represents how well samples from p_G are able to deceive the discriminator. In total, the complete network can be described by a two-player minimax game by the following value function [45]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}(x)[logD(x)] + \mathbb{E}_{z\sim p_{z(z)}}[log(1-D(G(z)))]$$

Goodfellow et al. further describe that, in practice, this game should be implemented using an numeric and iterative approach. To avoid overfitting of D, one must alternate between k optimization steps of D and one optimization step of G. Then, under the condition that G changes sufficiently slow, the results of D can be maintained near the optimal solution. Figure 4.3 visualizes the process of the generator and discriminator training "against" each other.

This figure shows the discriminator D (blue dashed line) training simultaneously with the generator G (green line). The dotted black line represents the distribution p_x of the dataset x. The green line associated with the generator shows the distribution p_g of the generative dataset. The blue line associated with the discriminator shows its distribution so that it discriminates between the black and green distributions. Lastly, the black line at the bottom represents the domain where samples of z are taken from and the line above is the domain of dataset x. The arrows show the mapping $x = G(z)$ of the function represented by the generator.

In step (a) the network is nearing convergence. p_G already shows a lot of similarity to p_{data} and D is able to mostly accurately classify whether a sample belong to p_{data} or p_G. Thereafter, in step (b), D has performed k training steps since (a) and is slowly converging towards D(x) = $\frac{p_{data(x)}}{p_{data(x)}+p_{G(x)}}$. Then, in step (c) the training step of G is done where the gradient of D has pushed $G(z)$ to an area of the domain where it is less likely to be classified as p_G and more likely to be classified as p_{data}. Finally, in step (d), p_G has become (nearly) identical to p_{data} and as a result, D is no longer able to discriminate

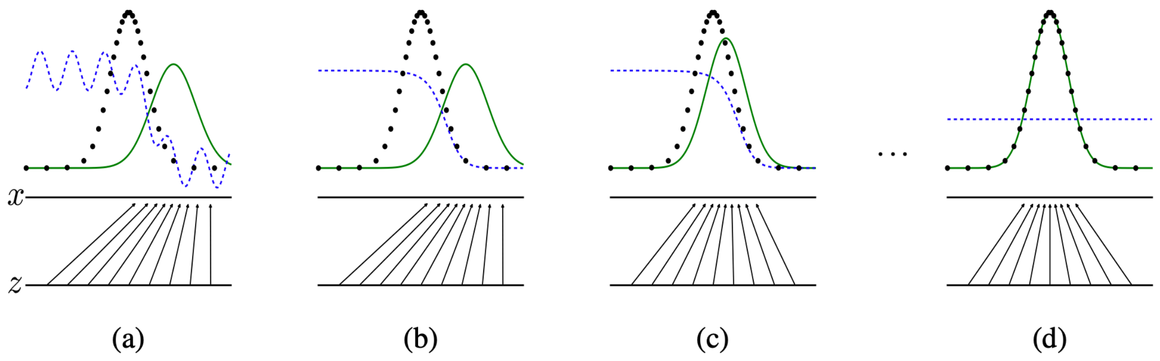


Figure 4.3: Convergence of the training process of an adversarial network [45]

between the two distributions, meaning that D is no longer being able to obtain a higher confidence for a sample x belonging to p_{data} than $1/2$.

Even though GANs pose a great deal of advantages over many other techniques of synthesizing a good representation of a dataset, there are still some big challenges to be addressed in the field. One of the most serious of these problems is called *mode-dropping* or *mode collapse*. Mode collapse is the phenomenon of multiple portions of the target probabilistic distribution being omitted from the probabilistic model produced by the GAN. In their work, Bau et al. [5] analyze this problem in the context of synthetic image generation with GANs. The authors find that the object classes seemingly ”forgotten” by the produced model are not simply distorted or noisy, but rather dropped entirely. As an example, they observe that when synthesizing images, large human figures and parallel lines in fences appear to be ”too hard” of a class for the GAN to learn and as a result are omitted entirely from the produced images. Despite this, the average produced image is still of high quality. Furthermore, it is a non-trivial problem to translate the generation of synthetic data from a continuous domain, for instance the image domain described above, to a domain consisting of a mix of continuous and discrete data [102]. In many domains, including the domain of financial fraud, the discrete columns in a tabular dataset are often unbalanced and of high cardinality. On top of this, continuous columns may have multiple varying modes. These factors combined cause existing approach to fail to properly model datasets in this domain. For this reason, the next section will explore and describe an approach to synthesize tabular data, which is built upon later in this work.

4.3.2. CTGAN

CTGAN or *Conditional Tabular GAN* is a framework that addresses the aforementioned challenges posed by GANs. In their work, Xu et al. [102] compare existing state-of-the-art techniques techniques such as Bayesian networks and deep neural networks using both real and synthetic datasets. Their results show that on tabular data, the synthetic data generated by these techniques fall short on various metrics like *machine learning efficacy* and *likelihood fitness*. The approach proposed by these authors, CTGAN, is shown to outperform these state-of-the-art methods on a good number of real and simulated datasets. Because this approach is applied in the contributions made by this work, this section will give a detailed overview of how CTGAN works, how it performs compared to other methods of generating synthetic tabular data and why it is applicable in the field of financial fraud detection.

The process that CTGAN follows to learn the distribution of a target dataset consists of 2 steps: *mode-specific normalization* and a *conditional training process*. Furthermore, the conditional training process consists of three key elements: the *conditional vector*, the *generator loss* and the *training-by-sampling* method.

Mode-specific Normalization

The first way in which CTGAN distinguishes itself from a normal GAN is how the data from the target dataset is represented. As described earlier, a regular GAN uses a minimax game to normalize continuous values. To account for (unbalanced) discrete columns and columns with an intricate distribution. The authors divide this step into 3 sub-steps:

1. Each column in the tabular dataset is considered separately and independently. For every continuous column C_i, a *variational Gaussian Mixture model* (VGM) is used to estimate the number of modes in the distribution of the values in the column. On top of that, a Gaussian mixture, a function comprising of multiple Gaussian distributions, is fitted onto the data.

2. Compute the probability of coming from each mode found in the previous step for each value $c_{i,j}$ in C_i.

3. Sample one mode given the joint probability density of the Gaussian mixture and use this mode to normalize $c_{i,j}$. If, for instance, 3 modes were found and the second mode is sampled, $c_{i,j}$ is now represented as the one-hot vector $\beta_{i,j} = [0, 1, 0]$. This normalized value then becomes $\alpha_{i,j} = \frac{c_{i,j}-\eta_2}{4\phi_2}$. To clarify, η_2 and ϕ_2 are the mode and the standard deviation of the second mode respectively.

As an example, assume we have a tabular dataset consisting of a continuous value, followed by a discrete value, followed by one last continuous value. The dataset is now represented in the following mode-specific normalized way:

$$r_j = a_{1,j} \oplus \beta_{1,j} \oplus d_{1,j} \oplus a_{2,j} \oplus \beta_{2,j}$$

Here, $d_{i,j}$ represents a one-hot encoded discrete value. For discrete values, making a one-hot encoded vector is relatively simple: if there exist n unique values in the column, the one-hot vector is a sparse vector of length n with value 1 in the location of the vector associated with the specific discrete value.

In the standard GAN described earlier, in each generator step the generator G is fed a sample from some multivariate distribution. One problem with taking random samples, is that if a certain discrete column is sufficiently *unbalanced*, this specific category or class is not adequately represented during the training process. The problem of *class imbalance* was discussed earlier in the context of supervised learning, but has even more impact in this context, since multiple imbalanced columns can exist in a tabular dataset. Furthermore, one is not able to tamper with the data in the dataset, since the row-domain distribution needs to be kept intact. The CTGAN approach circumvent these problems by using the three key elements mentioned earlier.

Conditional vectors

Conditional vectors are used to represent the collection pf discrete columns per row. The goal is to efficiently and evenly sample values from columns during the training process. To this end, consider the value k^* from the discrete column D_{i^*} As briefly discussed earlier, every categorical column $D_1, ..., D_n$ is transformed into a one-hot encoded vector $d_1, ..., d_n$. The authors now transform each of these one-hot vectors into a so-called *mask vector*. Based on what condition we want to represent, described by the condition $D_i = k^*$, and the ith one-hot vector d_i, which is equivalent to $[d_i^{(k)}]$ for $k = 1, ..., |D_i|$, the authors construct a mask vector in the following way [102]:

$$m_i(x) = \begin{cases} 1 & \text{if } i = i^* \text{ and } k = k^* \\ 0, & \text{otherwise} \end{cases}$$

Then, the conditional vector is the concatenation of $m_i * (k)$ for $k = 1, ..., |D_i|$. For example, consider a tabular dataset with two discrete columns: $D_1 = \{1, 2, 3\}, D_2 = \{1, 2\}$. If one now wants to represent the condition $D_2 = 2$, the following mask vectors are constructed from the columns: $m_1 = [0, 0, 0]$ and $m_2 = [0, 1]$. The conditional vector then becomes $[0, 0, 0, 0, 1]$. Using these conditional vectors, the generator is able to interpret the conditional distribution in the row-domain based on particular values from discrete columns.

Generator loss

Based on the various conditions in the form of $D_{i^*} = k^*$, the generator is able to produce a wide range of one-hot vectors for discrete columns. To push the conditional generator in the direction of generating discrete vectors that resemble the mask vectors obtained from the original dataset as much as possible, the authors introduce *generator loss* in the form of cross-entropy between mask vector m_{i^*} and the generated discrete vector $\hat{d}_{i^*}$.

Training-by-sampling
With conditional vectors and generator loss under their belt, the authors define the training process as a set of steps. Over time the generator learns to fit its generated values in the form of the conditional distribution $P_G(row|cond)$, *cond* being the conditional vector, to the target conditional distribution $P_G(row|cond)$. The discriminator in turn assesses these generated vectors making use of the $cond$ vector and the generator loss to estimate the distance. Gradually this allows for the network to explore all possible values in the discrete columns. The steps used for training the network are roughly:

1. Construct N_d zero-vectors as mask vectors, corresponding to each discrete column in the table;
2. Select a random column from the N_d vectors;
3. For this column, create a probability mass function (PMF) across the range of all its values;
4. Select a value k^* based on this PMF;
5. Set the value in the corresponding mask vector to 1; and
6. Calculate the $cond$ vector as described earlier.

Using this method to construct evenly (not uniformly) distributed row samples, the generator is pushed in the direction of reconstructing the distribution of the original dataset: $P_g(row|D_{i^*} = k^*) = P(row|D_{i^*} = k^*)$. In turn, this allows one to reconstruct a row from the original distribution, using Bayes' theorem:

$$P(row) = \sum_{k \in D_{i^*}} P_G(row|D_{i^*} = k^*)P(D_{i^*} = k)$$

5 Literature Research: Privacy and Ethics in Automated Fraud Detection

Ever since the Facebook-Cambridge Analytica scandal, where sensitive personal information of about 50 million Facebook users was leaked in a major data breach [15], the public awareness of data privacy has increased significantly. E-mails, invoices, contracts and bank transfers, among many other types sensitive personal information, were used to exploit people for financial and political gain. Of course, Facebook is not the only culprit. The explosive growth of social media and various online service platforms has lead to a significant amount of personal, and often sensitive, data being stored of billions of users. As the Facebook scandal has proven, many of these technologies are of a pervasive nature and as a result pose a multitude of threats to the privacy of the individual. Examples of such threats are unwanted surveillance, neglectful disclosure of private data or the unauthorised use of personal data by third party companies [24].

Nowadays, the field of finance has significant overlap with many of the aforementioned companies. As the efficiency and scale of the field increases due to the application of state-of-the-art technologies, so do the challenges in the field of ethics and privacy. If we consider the sub-field of transaction monitoring, which this work focuses on, we find that the amount of sensitive data kept on users has soared. As the advent of modern technologies in finance has opened many new doors for fraudsters, increased regulations have been imposed by the DNB in terms of record-keeping and data retention [23]. Vast amounts of different types of user-related, and often sensitive, data have to be retained in order to comply to these restrictions. Examples are identity documents, conversations with the customer and transactions made by the customer. On top of this, many state-of-the-art techniques for fraud detection, for instance supervised learning techniques, require a tremendous amount of training data in order to learn how to make meaningful predictions. As the quality and application of such intricate automated fraud detection systems grow, the need for model designers to factor in the ethical side of their models becomes more apparent. While these systems may show an overall strong performance, no system is perfect. Mistakes made by these systems potentially have a huge impact on an individuals financial security or the reputation of a financial institution. Especially for ”black box” systems like machine learning models trained on enormous datasets there is the need to both explain and substantiate the choices made by the system.

This section will provide an overview of how privacy and ethics relate to data mining and machine learning in the field of financial fraud detection. First, we explore the types of mistakes made by automated fraud detection systems along with their respective implications. Then, a clear definition of privacy in the context of web data mining is given, along with a description of the unique relationship between the two. Afterwards, various techniques to achieve a level of guaranteed privacy in data mining are discussed briefly, after which the method that is partially employed in this work is described in detail. Lastly, an argument is made for why this specific method is suited for the field of financial fraud detection.

5.1. Mistakes in fraud detection

While automated fraud detection systems have seen a drastic improvement over the past years, the fraud detection industry is still plagued by a significant margin of error. Where the goal of an automated fraud system is to detect as many fraudulent transactions as possible, the trade-off between reducing the number of fraudulent transactions flying under the radar, *false negatives*, and the number of transactions incorrectly classified as fraudulent, *false positives*, is very delicate. Beneish & Vorst have investigated this trade-off and define 4 challenges that system designers and decision makers generally face when working with this trade-off [6]:

- **Costs and benefits analysis:** thus far, the estimates of the trade-off between the costs and benefits of both false positives and false negatives has remained in the realm of assumptions;

- **Unrealistic assumptions:** the most commonly used metrics to compare the performance of a fraud detection system generally make unrealistic assumptions about the relative costs of erroneous predictions;

- **Costs incurred:** increasing the performance of the model at the detection of fraudulent transactions, in other words decreasing the false negatives, comes at the cost of an increase in the number of false positives; and

- **Misrepresented insights:** because one needs human expert investigation to judge the performance of a fraud detection system, extreme examples often steer the insights found to be misrepresentative of the costs and benefits of the full picture. On top of this, different stakeholders involved usually have different concepts of the costs and benefits associated with the system, further increasing the degree of misrepresentation of the found insights.

In essence, these challenges create the need for a system designer to use his or her knowledge about the domain and the specific use case to determine not only a fitting set of metrics able to accurately represent the aforementioned trade-off, but also aim to find such a trade-off that satisfies the costs and benefits of the entire group of stakeholders. Examples of such stakeholders, as defined by Beneish & Vorst, are *auditiors*, *investors* and *regulators*, officials from instantions responsible for fraud regulations . Here, auditors are the persons in possession of all knowledge related to the fraud detection system and responsible for its implementation. Investors are the people putting their funds on the line wit the expectation of the return of profit after due time. Lastly, regulators are the people responsible for defining the rules of operating such a system, as well as regulating the respective market(s). The differences in goals regarding the performance of the automated fraud detection system, ranging from turning maximum profit to running a sufficiently ethical operation, makes accurately assessing its quality an arduous task. To motivate many of the design choices made in this work, this section will go into more detail about the implications of both false positives and false negatives, from the perspective of the aforementioned challenges and stakeholders.

5.1.1. False negatives

A false negative essentially means that a fraudster gets away unnoticed after committing fraud. The consequences of false negatives are, unfortunately, colourful in nature, ranging from providing means for terrorist funding to allowing users to be duped out of significant amounts of money. In the field of financial fraud, false negatives are often significantly more costly than false positives.

To auditors, the main goal of the system is the prevention of false negatives. From their point of view, the costs associated with false negatives include, but are not limited to: litigation, reputation costs and the foregone profits accompanying the loss of the odd client as the result of the public revelations of the auditors' shortcomings. To auditors, these costs often significantly outweigh the costs associated with false positives. To investors, the costs of false negatives are expressed as the loss of value after the reveal of the fraud. As an extension, costs of false negatives directly translate to financial losses for the users of the financial institution. Lastly, regulators are concerned about the value loss incurred by fraud to the market as a whole, as well as it's threat to the financial well-being of the users of the financial institution [6].

5.1.2. False positives

A false positive in this context means a user wrongly gets classified as a fraudster. This has 2 major consequences: first and foremost, depending on the direct consequences, the user might (temporarily) lose access to his or her monetary account(s), suffer from reputational loss and in turn is likely to lose trust in the institution. Secondly, false positives are costly in terms of extra, otherwise unnecessary man-hours needed to manually review a flagged transaction. To auditors, false positives mean investing more resources into audits (for which a client ethically should not be billed) and potential loss in income resulting from a client resigning. To investors, the costs of false positives stem from the profit foregone by not investing in clients marked as false positive, as well as the possible increment in audit fees paid by users. Then, to regulators, costs incurred by false positives lead to average loss of value of the market as a whole when users or companies are publicly mis-labeled as fraudsters [6].

To conclude: we find that in the literature various models exist that define a cost ratio between false positives and false negatives. The most significant common denominator between these models is that the costs of false negatives often significantly outweigh those of false positives. As an example, in a prominent research done by Dechow et al. [25] a model to score the costs incurred by financial fraud is developed, based on three alternative, but similar models. These models assume a 143:1, 86:1 and 82:1 cost ratio between false negatives and false positives respectively.

5.2. Web data mining and privacy

Defining privacy

To understand the relationship between the mining of web data and *privacy*, we first need a clear definition of privacy in this context. The general notion of privacy has been defined in various ways over the past centuries. For instance, in 1890 Warren and Brandeis defined privacy as ”the right to be let alone” [12], whereas, in 2017, Parent defines privacy as ”the condition of not having undocumented personal knowledge about one possessed by others” [75]. As can be seen from these examples, the definition of privacy has progressed towards a setting where (sensitive) personal information is regarded as a commodity to which an individual has a right of secrecy. In line with this remark, in the context of the (social) web we consider privacy in the form of *informational privacy*. From the point of view of the user of a web service, Tavani characterizes informational privacy by stating two main concerns [96]:

1. The awareness and knowledge of the user about whom information is collected; and

2. How the information about the user is used.

Defining web data mining

With the advent of data mining, especially various machine learning techniques, seeing increased application across many online services, the need for proper implementations guaranteeing privacy, fairness, accountability and transparency has become of increased concern. In their work, Kosala and Blockeel divide data mining of web data into three areas of interest, based on the part of the web that the data miner chooses to mine [58]:

1. **Web content mining:** the identification of useful information though mining of documents, data or other content found on the web. Examples of such data types are pictures, hyperlinks and audio;

2. **Web structure mining:** the process of constructing a model of the linked underlying structures of the web. Often used to discover similarities and relationships between web pages; and

3. **Web usage mining:** the process of making sense and discovering patterns in the data produced by people using the web or a specific web service.

Considering these definitions, we shift our focus towards the last of these areas, as our source of data is entirely dependant on the behavior of the users of a certain web system. As opposed to web content mining, web usage mining focuses on mining so-called *secondary* data, which is produced by users interacting with a web service.

5.2.1. Web data and threats to privacy

Firstly, the most direct and obvious threat to ones informational privacy is when a users personal data collected through web services becomes public. There are many ways in which such data could reach public accessibility, such as data leakages and unauthorized access or data sharing for open-sourced data mining purposes. When records kept on individual users contain sensitive data, for instance medical records, financial information or residential data, such data publications are a major breach of ones informational privacy. To counter this, organisations will often encrypt or remove such sensitive data before the data is made public with the assumption that privacy is now preserved. In most cases though, the remaining data is still sufficient to re-identify individuals in the dataset [95].

Secondly, considering the two concerns about informational privacy defined by Tavani, it becomes evident that a large majority of data mining applications fall short. While the first concern usually holds true and the user is aware that his or her information is being collected, the user still in most cases has little to no means to find out how the information is being used. This discrepancy is a direct result of the complexity of the data mining application to a regular user, both in terms of understanding how the data mining algorithm works and how the algorithm uses the users data. Furthermore, the data mined from large datasets will produce new results such as user profiles or patterns. The normative protections that are in place to ensure the protection of personal and sensitive data often fall short in also protecting this ’secondary’ type of personal data [96]. As a result, many data mining activities do not directly violate the informational privacy of an individual, while still causing a threat to it. Examples and implications of such indirect threats will be discussed in the next subsection.

5.2.2. Sensitive data and societal responsibilities

As many societies gradually become more digitized, personal information, including sensitive information, is becoming an increasingly important resource to a plethora of web applications and services. Sensitive characteristics, such as the nationality or the gender of a user, have become an increasingly delicate subject over the past years. Bias, exclusion or scrutiny towards a person based on factors he or she cannot control have become a major factor of criticism in many online activities nowadays. Consequently, the GDPR has begun qualifying certain user-related attributes to be sensitive. This includes political preferences, ethnic background, religious views, trade union memberships health data, sexual data and genetic or biometric data [99]. Data mining algorithms that are trained on data produced by users of a web service are prone to bias resulting from the dataset, which, in the end, is still a result of social behavior. This may lead to automated decision makers overestimating the matter in which certain, often sensitive, attributes impact the decision being made. Considering Tavani’s second concern for informational privacy, no reasonable user would condone decisions being biased towards such sensitive attributes. Offenses such as fraud or intrusion should be inferred from objective misbehavior, rather than being biased towards uncontrollable attributes such as race, gender or religion [47]. No longer should model designers only pay attention to **what** results their decision making algorithms produce, but also **how** these results are obtained.

5.2.3. Attacks on privacy

Furthermore, not only do organisations owe their users the responsibility of their automated decision makers making unbiased choices, they have to ensure safekeeping of the (sensitive) data that is collected of their users and their behavior. As mentioned, oftentimes even sanitized or encrypted datasets contain enough information for a potential adversary to still identify individuals and their sensitive information. An example of one such case is the user dataset that was made publicly available by Netflix for their movie recommender-system contest. Even though this dataset contained only anonymized user data, it was shown that it was still possible to re-identify sensitive information of their users like their sexual orientation, leading to a slew of lawsuits [92]. As a result, *Privacy Preserving Data Publishing* (PPDP) has become an emerging research topic in the field of data mining. The general idea behind PPDP is to be able to safely publish and store large datasets for data mining purposes, while minimizing the loss in data mining utility. This is often achieved by anonymizing the dataset to the point where individual records to be indistinguishable from one another, thus providing a privacy guarantee for an individual and his or her sensitive personal data.

PPDP techniques cover a wide range of approaches. including but not limited to heuristic-based techniques, differential privacy and reconstruction-based privacy. The problem of optimally sanitizing a dataset has been classified as an NP-hard problem and as such Heuristics can be applied to approach such an optimal solution [96]. An example of such an approach is k-anonymity, where we take a set of anonymization steps in order for each record to be indistinguishable from $k-1$ other records. Here, again, we trade data mining utility for privacy preservation. Even though k-anonymity has been proven to provide a fairly good privacy guarantee, it is still vulnerable to attacks such as a *homogeneity attack* and *background knowledge attack*, where adversaries predict sensitive attributes of each k records by using statistical methods and external background knowledge respectively [106]. Differential privacy is a method that aims to achieve a high degree of privacy, at a minimal loss of data mining utility. It ensures that adding or removing a single item from a dataset does not significantly affect the data mining utility. To achieve this, a non-deterministic function F takes the original data set D_1 as input and adds statistically sampled noise to produce a differentially private dataset D_2, which differs on at most one element [33]. This method allows the data miner or model designer to still gain useful insights about the group as a whole, while disallowing an adversary to learn meaningful information about an individual. Lastly, reconstruction based methods make use of randomization or cryptography to alter sensitive data fields, which are reassembled upon use.

5.3. Privacy and synthetic data

The sections above have discussed collecting, storing and publishing sensitive data collected of users using a web application. While these subjects have proven to be an interesting challenge in terms of guaranteeing a certain degree of privacy, using this data to train a machine learning model imposes further challenges in the field of privacy. Large-capacity machine learning models that are trained on a dataset that contain sensitive fields have shown to remember large amounts of information about the users present in the training dataset. As a direct result, these models are at risk of leaking personal information about users [103]. To combat this and the aforementioned privacy threats, one can employ synthetic data as (a part of) the training dataset. This section will go into more detail about the privacy implications of using synthetic data for data mining and, more specifically, machine learning purposes.

5.3.1. Advantages

When considering synthetic data from a high level point of view, one could conclude that synthetic data fully protects the privacy of the individuals in the original dataset. After all, none of the records in the synthetic dataset represent an actual person, so how would an adversary ever learn anything about a real person from this dataset? Unfortunately, in reality it is not that simple. Many factors related to the quality of the synthetic data play a role in its preservation of privacy, as well as the machine learning utility that the synthetic data provides. When considering privacy preservation at the level of storage, publication and usage of training data, synthetic data has the following advantageous properties:

- **Data storage:** according to the *General Data Protection Regulation* (GDPR), parties holding user data containing personal information have to handle this data in a strictly confidential way. Over the past years there have been numerous cases of data leakage of large institutions. Whether this was through hackers breaking into and publishing datasets or accidental disclosures, the result is the same: user data with personal information is now publicly available. To mitigate this risk, one could opt to replace the training data with synthetic data. This has two distinct advantages in terms of storage:
  - Synthetic data does not contain any direct personal information about individuals. As a result, no individual can be directly harmed by the publication of the dataset. A clever adversary might still be able to determine *some* sensitive information from the synthetic data though, which will be discussed later in this section.
  - In many cases, there is no need to even store the synthetic records. Only the trained model to generate the synthetic records has to be stored. As a result, significantly less storage for training data is need as an infinite number of synthetic records can be generated ad-hoc [103]. On top of this, in case of a data leak, an adversary would need a sufficient skill set to obtain information from the model, further increasing the privacy preservation of the system.
- **Data publication:** as mentioned before, direct publication of dataset containing (sensitive) user data is very likely to violate the privacy of an individual. As a solution, one could opt to only publish the statistical properties of the fields in the dataset, thus protecting the individual, although this

comes at a great cost in terms of loss of data value. By publishing a high-quality synthetic dataset that sufficiently represents the real dataset, we allow for a high preservation of privacy while still allowing researchers and data miners to create high-quality models.

- **Data usage:**
  - Companies holding large amounts of user data containing personal data used for data mining purposes not only need to comply to the confidentiality rules imposed by the GDPR when considering data storage, but for data usage as well. Many software developers at such a company will at some point have to access this data for model training or bug fixing. In most cases, companies do not want their employees to have direct access to real user data, as this would count as a violation of an individuals privacy. Providing employees with synthetic data that preserves relationships found in the original data, while removing all direct sensitive data removes this barrier [103]. As an example, at Uber allowed employees to gain access to real customer data, resulting in a settlement with the US Federal Trade Commission [49]. Uber now employs a system for accessing perturbed customer data.
  - No real data is used to train the model and thus to generate predictions. This fact removes the need to be cautious about the machine learning model leaking potentially sensitive personal data, for instance in a scenario where requests can be made to a trained model through a publicly available API.

Of course, such advantages in privacy preservation do not come for free. Much like many of the aforementioned privacy preserving techniques, employing synthetic data comes at the potential loss of machine learning utility. This trade-off is explored in detail in experiment section 12.2. Lastly, to gain a concrete understanding of how privacy relates to synthetically generated data, Hittmeir et al. [50] introduce a method to measure the degree of privacy that a synthetic dataset guarantees. Their method entails calculating the nearest neighbour distance, in this case the Euclidian distance, between the training dataset and the generated dataset. This way, similarities or even rows containing literal data from the training set can be detected and quantified. In an ideal scenario, the distance between the generated dataset is sufficiently large, while the machine learning utility of the dataset remains high, in the best case equal to that of the original training data. This method and more privacy preservation metrics that are used in this work are introduced and explained in method section 11.3.

5.3.2. Attacks on synthetic data

As the previous section has shown, synthetic data provides various advantages to preserve privacy in a training dataset. Even though synthetic data might seem a completely separate entity compared to the real dataset, a shrewd adversary still poses a threat to the privacy of the individuals in the original dataset. Because the synthetic records are samples from a joint distribution modeled after the real dataset, statistical techniques exist to determine sensitive information in the real dataset, based on the relationships found in the synthetic dataset. Such an attack is an example of the aforementioned background knowledge attack. Method section 11.3.2 describes one such attack strategy and performs this attack on a synthetic dataset created from a real-life dataset with the goal of providing another angle of the degree of privacy preserved by this synthetic dataset.

6 Case-study: Automated Fraud Detection

As a stepping stone for this work to take shape, a clear conception of real-world automated fraud detection has to be build up from the ground. To achieve this, over the thesis project duration, various work was done on the front line of automated fraud detection: a rule based system. Implementing and adapting transaction monitoring rules requires one to understand fraudulent behavior through fraud analysis, transaction monitoring rule design, back-end development and end-to-end testing. In other words, the entire process provides all the necessary background knowledge to in turn take automated fraud detection to the next level.

This section will provide the reader with a ground-up overview of the traditional way of detecting fraudulent transactions in tandem with the contributions made to this system at bunq. First, an abstract definition of machine learning predictions with respect to fraudulent transactions is given in the form of the relationship between human expert knowledge, automated systems and the notion of unknown unknowns as described in section 4.2. Afterwards, this relationship is linked to existing approaches to combat fraudulent behavior. Then, the contributions made to the rule-based system are used as examples of such approaches. Lastly, we discuss the shortcomings such a relatively simple system for fraud detection. The goal of this chapter is to introduce the reader to automated fraud detection, as well as motivate the use of a weakly-supervised machine learning approach as introduced in this work.

6.1. Unknown unknowns in fraud detection

As thoroughly discussed in section 4.2, unknown unknowns represent cases where an automated decision maker makes a mistake with a high confidence. In other words, it does not know that it has made a mistake. Because the knowledge held by the automated decision maker is fully dependant on the training data that it is supplied with, which in turn is dependant on human experts providing labeled data, it becomes clear that UUs are a blind spots to both the automated decision maker and the human expert team behind it. It is as the Johari window explains: this knowledge is part of the *area of unknown activity*, in other words, this information is not known to anyone. Yet, we can assume that information does exist. Through extensive analysis of transactions, one aims to somehow eventually discover this information and as a result, fraudsters will always be at least one step ahead.

When considering binary classifiers, be it a simple transaction monitoring rule or an intricate machine learning model, we are able to categorise the relationship between the potential fraudulent nature of a transaction and the classification results of the automated decision maker with respect to the Johari window. In such a setting, we consider the knowledge held by the model from the perspective of ourselves: the fraud analysts, annotators and model designers. We then define the Johari window as follows:

- A highly confident correct prediction corresponds to a **known known**: the right information is **known** to us and as a result has been correctly represented in the training data leading to the information being **known** to the model, as indicated by the high confidence score.
- **Known unknown**s occur in scenarios where certain patterns and categories of fraud are **known** to us, but (for the time being) **unknown** to the model. Such situations occur whenever certain

types of fraud become known to us, for instance through extensive analysis or external information. Known unknowns also result from known shortcomings of the model, for which many approaches usually exist to remedy them as explained in section 4.2. One can turn known unknowns into known knowns by applying said methods or collecting ample training data on the newly discovered fraud type. This training data can be collected for instance through new transaction monitoring rules or retroactive labeling of the training dataset.

- Following the same logic, **unknown known**s are instances where certain information is **known** to the model, that is **unknown** to us. An example of a known unknown is a pattern recognized by the model to be a strong indicator of fraud, that we have yet to recognize due to its underlying nature. Through sufficient examples and effective explainability methods one is able to extract this knowledge from the model and turn unknown knowns into known knowns.
- Lastly, **unknown unknowns** are instances where the model makes a high confidence, incorrect prediction. At the time of the prediction, its incorrectness is not known to us. As discussed in section 4.2, UUs are the result of a mismatch between the training data and the data found after the model is deployed. Often, this mismatch is the result of a class being underrepresented in the training dataset. Furthermore, a shift in population can also lead to this mismatch. Such a shift is not hard to imagine in the field of fraud prevention, as fraudsters are always changing their methods to fool both automated systems and human experts. On top of this, as we learned from section 5.1, false negatives are significantly more costly to financial institutions than false positives. For these three reasons we define the following theorem:

  Theorem 1 *If transaction t is considered to be an unknown unknown to model M, t is an error of the category false negative made by model M.*

  This theorem does not exclude the possibility of UUs in the form of a false positive to exist, but due to the fact that such instances are considered to significantly more rare and significantly more cost-incurring compared to their false positive counterparts, we assume the theorem above.

In essence, the goal of this work is to transform unknown unknowns into known knowns. In order to achieve this, we aim to first transform unknown unknowns into known unknowns by teaching the model to be more robust against previously unknown fraudulent instances, with little to no human intervention. This method will be described great detail in the upcoming method chapters and explored thoroughly in experiment 12.4.

6.1.1. Shortcomings of a rule-based system

Now that we have a better idea of how transaction monitoring rules generally work and perform, we will discuss the main shortcomings of using this method for automated fraud detection.

Labour intensive

The process of creating and maintaining monitoring rules is excessively labour intensive. The creation process of a rule is arduous, consisting of fraud analysis, design, implementation, end-to-end testing and performance analysis. On top of this, many different stakeholders are present during this process, ranging from risk officers analysing the trade-offs implied by the rule, to data scientists evaluating the costs and benefits of implementing a rule. Furthermore, as rules are static by nature, the upkeep of rules is also a noteworthy aspect in terms of labour required. The strategies and approaches of fraudsters are ever changing. For this reason, keeping an eye on the performance of a rule, making adjustments or improvements and going through the entire legal process again for each change makes keeping rules up to date with the current fraud environment a tremendously arduous task.

Large number of false positives

Monitoring rules tend to be aimed at one specific category or instance of fraud. The ultimate goal of any transaction monitoring rule is to catch any fraudulent scheme corresponding to its respective category. This creates a trade-off for each rule: does one create a fairly broad rule which catches the majority of a type of fraudulent behavior at the cost of a high number of false positives or does one create a very strict rule which has a high true positive rate, at the cost of many missed fraudulent cases (false negatives)? The former creates many man-hours of needlessly investigating innocent transactions, whereas the latter results in a high demand for extra similar rules to also catch niche cases of the same fraud category, thus also requiring many development hours.

Limited applicability
While rules provide a fairly strong short-term solution to certain fraudulent behaviors, this specificity is also a disadvantage. As explained in chapter 3, the field of financial fraud is unfortunately very broad of nature. Especially nowadays with the advent of online banking, the ways in which a fraudster is able to commit fraud are plentiful. Monitoring rules that combat a certain type of fraud usually have little to no applicability on other types of fraud, even if they are slightly unrelated.

Rapid obsolescence
When fraudsters inevitably adapt their scheme, the static nature of rules disallows them to adapt without human intervention. Such intervention requires another iteration of the lengthy process described above. Due to the high amount of money usually involved in fraudulent schemes, fraudsters tend to be on top of their game. In some cases a monitoring rule can even become obsolete the same day that it was deployed, due to clever fraudsters reverse-engineering the simple rule simply by trial and error.

7 Method: Feature Value Attribution Model

Moving on from transaction monitoring rules, we consider what lies at the core of the research done in this work: the supervised machine learning model used to distinguish fraudulent transactions from legit ones, which we will call the *Fraud Detection Model* or *FDM* from now on. The end goal of this work is to improve an existing FDM in terms of generalizability and robustness against fraudulent transactions that would otherwise go unnoticed. In other words: to decrease the number of false positives created by the FDM. Since the majority of the contributions made by this work are in the form of augmentations to the training dataset, the existing FDM itself remains mostly unchanged.

The FDM takes a tabular dataset as its training data. Each row in this dataset corresponds to a single transaction made by a user. This dataset consists of numerical, categorical and boolean feature types. Examples of numerical features are the amount of the transaction, the age of the user or the time since the user signed up to the service. Examples of categorical features are the nationality of the user, the type of the transaction or the country of the receiving end. Lastly, examples of boolean features are whether the amount of the transaction is a multiple of 10, whether the transaction is international or if the transaction is related to a certain market (for instance gambling). Based on such a row of features, the FDM outputs a probability for a new transaction to be fraudulent or legit. A threshold for such a transaction to be considered fraudulent or legit is often obtained based on a specific performance metric such as F1 score or the area-under-the-curve (AUC). Section **??** goes into detail of how we implement such a FDM.

One of the main contributions that this work makes is the intuitive way in which a supervised learning model in the form of a decision tree ensemble, along with global Shapley values are used to deduce meaningful insights on the effects of certain feature values. We will call this model the *feature value attribution model* or *FVAM*. By training a supervised model on a dataset that uses target labels to indicate whether a certain type of fraud is an UU to the fraud detection model, we are able to construct an explanation model that helps us understand which (combinations of) feature values push the model towards miss-classifying certain categories of fraudulent transactions. In the field of fraud detection, this knowledge is alluring as fraudsters are constantly trying to find new ways of manipulating the characteristics of their transactions in order to trick automated fraud detection systems. Hence, the goal of this section is to create a technique to quantify the degree in which features can be manipulated to commit fraudulent transactions without being detected by the fraud detection model.

7.1. Training dataset

The training dataset that we use to train the FVAM is very similar to the FDM. To train the FVAM we use the exact same set of features as for the FDM. The only, but very significant, difference is the target label. Instead of the binary classification scenario of the FDM where we use the label 0 for legit transactions and 1 for fraudulent transactions, we flip the label based on whether the transaction is a false negative of the FDM. As mentioned in the previous section, after a transaction has been classified by either the FDM, through the rule-based system or external means, we are able to obtain our ground truth label. Obtaining the target label of a transaction does however **not** mean that this category of transactions is now a known unknown. Let it be clear that obtaining the target label of a single or multiple transactions does not imply that the model or us as the model designers are able to grasp the full picture of a certain category of fraud. This label, together with the predicted label by the FDM, allows us to group every existing transaction into two categories:

1. **False negatives:** the class that we are most interested in. A false negative is found when the ground truth label of a transaction is, through any means, found out to be fraudulent (target label 1) **and** the FDM has predicted the transaction to be legit (class label 0).
2. **Other:** every other relationship between the ground truth and model labels. This includes correctly classified transactions, both fraudulent and non-fraudulent cases. This also includes **false positives**, where the model has predicted 1, but the actual ground truth label was 0. While this type of mistake is also not desirable, we remember theorem 1 and label our training dataset accordingly.

More formally, the dataset labels result from the mapping m:

$$m: \{0,1\} \rightarrow \{0,1\} \text{ such that}$$
$$\begin{cases} m(label_i) = 1 & \text{if } label_i = 1 \text{ and } p_{model} < threshold \\ m(label_i) = 0 & \text{else} \end{cases}$$

Here, $label_i$ is the target label of row i in the training dataset, which always corresponds to the ground truth label of the FDM training dataset. p_{model} is the predicted probability by the FDM whether this specific transaction is fraud. If this probability is higher than the threshold, we consider a transaction to be classified as fraud and vice-versa. While the actual predictions of this model are of little interest to us, the global Shapley values representing the feature value attributions provide us with very valuable insights, which will be discussed in detail in the next section.

7.2. Implications of Shapley values

As explained in detail in section 7.2, global Shapley values describe how certain feature values push the model towards making particular decisions. In other words, these values allow us to find out the marginal contributions for all feature value combinations found in the training dataset (corresponding to the coalitions in the game theoretic context). For the FDM this is fairly straightforward: feature values with a high Shapley value push the model towards deciding whether a transaction has a high probability of being fraud, towards a value of 1. Vice-versa, feature values with a low Shapley value push the model towards deciding a certain transaction has a low probability of being fraudulent, towards a value of 0. For the FVAM however, interpreting the Shapley values requires an a more intricate thought process. We distinguish between 3 categories of features, based on their global Shapley values found across the training dataset:

1. Feature values with **large positive Shapley values** imply that the feature value in general pushes the FDM towards towards making **false negative** predictions. In other words, towards **missing fraudulent cases**. From here on out, we categorise these features as **prone to being exploited by an adversary**.
2. Feature values with **large negative Shapley values** mean that the feature value in general pushes the FDM towards **catching fraudulent cases** or **making a false positive** prediction. These features push the FVAM towards predicting 0, which as described above represents either a correctly predicted fraudulent transaction or a false positive, meaning these feature values on average push the model away from missing fraudulent cases. This does not imply that these features are ”better” than the features with large positive Shapley values, as these features may cause a significant amount of false positives.
3. Feature values with **zero-value Shapley values** mean that mean that the feature value in general has very little to no impact on the decisions made by the model. In short, this indicates that the feature is not able to convey enough relevant information to the model to base its predictions on.

We state the following theorem:

Theorem 2 *If feature f of the FVAM has a large positive global Shapley value, then large values of feature f push the model towards missing fraudulent transactions.*

This theorem provides an interesting insight, as this, combined with domain knowledge, helps anti-fraud measure designers understand how fraudsters are potentially adapting their strategies to beat automated anti-fraud measures. Of course, such patterns could also be learned by the FDM given enough time. However, this would require for many more fraudulent transactions of this category to be detected by through another system or external means, being annotated by a human expert, added to the new training dataset and the re-training of the FDM. The largest distinction between this approach and the FVAM approach, is that through the FVAM approach we are able to detect such patterns earlier and in a data-driven way, without the need for direct external input. The implications of theorem 2 are used in chapter 9 to create adversarial examples.

8 Method: Unsupervised Learning

As explained in section 6, the field of automated fraud detection with supervised learning suffers from the existence of UUs as a result from a heavy class inbalance, as well as shifts in population. In an effort to address these problems, we aim to construct high quality synthetic samples. The usage of meaningful and high quality synthetic samples is expected to help address the class imbalance problem and, as we will discuss in chapter 9, the usage of synthetic samples with certain characteristics are expected to help to solve the shift in population problem. When constructing these samples to address UUs, we again refer back to theorem 1, which means that we only construct synthetic samples that represent fraudulent transactions. After all, this is the minority class associated with the most costly category of errors: false negatives. The first step towards generating synthetic fraud cases is the training process of the CTGAN. As explained in section 4.3, CTGANs have various desirable properties for the generation of synthetic data, especially in the context of fraud detection. The second step involves efficiently and effectively reconstructing the feature arrays of the synthetic samples, following the same set of rules as the ones we would find for any real transaction. This chapter will give a detailed overview of the proposed approach to efficiently train a CTGAN, generate synthetic fraudulent samples with the trained CTGAN and propose two methods to reconstruct a valid feature array from the generated samples.

8.1. Dataset pre-processing

The first step towards generating synthetic fraudulent samples is the creation of a suitable training dataset. Since we are working under the assumption that there is an existing trained FDM, the starting point of this dataset is the tabular dataset used to train this FDM. The focus of this section is to explain how we pre-process this training dataset into a suitable dataset to train a CTGAN.

The first step of the pre-processing process is to filter out all of the legit transactions in the dataset. The goal of the CTGAN is to be able to learn a joint distribution representing fraudulent transactions, so to this end we only use fraudulent samples in our training dataset. Furthermore, the CTGAN framework requires the model designer to make a distinction between continuous and discrete values, which we discuss below.

8.1.1. Numerical features

Numerical features require very little pre-processing steps. Any integer or floating point values are suitable as attribute values for the CTGAN. Missing values are handled depending on the nature of the feature. For instance, a missing value for a key feature, such as the amount of a transaction, would imply that the entire feature array is erroneous or corrupt. As a result, we drop the full feature. Furthermore, a missing value in a null-able field, for instance a feature dependant on the type of a transaction in a sample not belonging to that specific type, is set to 0.

8.1.2. Categorical features

The pre-processing of categorical features is a bit more complicated. As explained in section 4.3.2, CTGANs have the desirable property of handling unbalanced discrete columns, even when the distribution of the values in such a column is significantly intricate or unbalanced, through mode-specific normalization. As a result, the model designer is only left with the task of handling missing values. Again, similar to the numerical features, we will drop any row that has missing values for an indispensable feature, for instance the type of the transaction. It is clear to see that every transaction should have a type and is otherwise not valid and thus unwanted in our training dataset. Furthermore, missing values of null-able features are handled based on domain knowledge about what that specific feature represents. Some examples to illustrate are:

- A feature to describe the users gender could in some cases be missing, when the user has decided not to disclose this information. In such cases one would replace a missing value by a string which implicates this information such as $unknown$.
- A feature which describes the sector of industry a certain user company belongs to. When working with multiple types of users, it is possible for certain users to not correspond to a certain type. Features that are specific to these types would in such cases contain missing values. In these cases we introduce a new feature value, for instance $user\_not\_a\_company$, to indicate this.

8.1.3. Boolean features

Boolean features are very simple, as this these features can simply be regarded as a categorical feature with only two possible values. Missing values are handled again based on the nature of the feature, provided that the feature is null-able. In most cases this means missing values are set to $false$. An example of this is the feature $mastercardPaymentWithdrawal$, which indicates whether the transaction is a Mastercard withdrawal. For every transaction that is not a Mastercard transaction, this value of this feature is missing. We can safely set this feature to $false$, as this value accurately represents the knowledge the feature represents.

8.2. Feature selection

Selecting a suitable set of features to train the CTGAN on is an essential step of the process. In our context, we want to consider the data generated by the CTGAN to be representative of possible attacks (fraudulent transactions) made by a fraudster. The selected feature set for the CTGAN to train on is chosen accordingly.

8.2.1. Independent features

There is a limited set of feature values per transaction that are within the control of the fraudster. Furthermore, these features often shape the majority of the feature array of the transaction, as many other feature values depend on these values. We will call these features *independent features* from now on. Examples of independent features where the fraudster is able to control the values are the amount of the transaction or the country from which the transaction was made. Not all independent features are in direct control of the fraudster. Aggregated features are counted as independent features as well, but have to adhere to a different set of constraints, which will be discussed shortly. For our CTGAN model, it is only necessary to learn the joint distribution of these independent features.

8.2.2. Dependent features

We now consider *dependent features*. These features are out of the control of the adversary, as these features are either dependent on the transaction made or consist of historical data about the specific user. These features are filtered from the training dataset of the CTGAN, as it serves no purpose to generate these features. Reducing the number of features before the training process serves reduce the complexity of the model, simplify the training process and in turn the computational power and time needed for the process [85].

8.3. CTGAN

This work builds upon the use of a CTGAN to construct synthetic fraudulent samples. Quite some work has been done in the field of synthetic data generation, which has been explored in literature section 4.3.2. This section will provide the reader with our motivation for using a CTGAN, describe the hyperparameters used to train a CTGAN and motivate a hyperparameter tuning approach used later in this work to explore the generation process of high quality synthetic samples.

8.3.1. Motivation

To motivate our choice for this method of unsupervised synthetic data generation, we consider the advantages introduced by CTGAN in the context of supervised learning for transaction fraud detection. Firstly, as mentioned before, we are dealing with tabular data rather than just continuous data. Such datasets often suffer from the problems where the continuous columns may have multiple modes and categorical columns suffer from a significantly imbalanced class distribution. CTGAN has been shown to effectively be able to model this kind of data, producing high-quality synthetic tabular data with mixed types and complicated distributions. On two novel benchmarks, which will also be evaluated in section 12.1, CTGAN is the first deep learning method to outperform Bayesian methods [103]. Secondly, CTGAN has been shown to achieve the same F1 score and accuracy as similar methods, while having a significantly larger nearest neighbour distance to the training dataset. As discussed in section 5.3.2, obtaining an advantageous trade-off between privacy guarantee in the form of a sufficiently high euclidean distance to the original training data, while retaining machine learning utility is a very desirable property of synthetic data. Thirdly, Xu shows in his work that CTGANs are able to capture complex interactions among variables, much more so than preceding GAN implementations [103]. In the context of fraud detection, this is a very desirable characteristic. Fraudulent behavior generally follows patterns that are a complex combination of various feature values. Compared to other fields, for instance image generation, such patterns are significantly more difficult to detect by human experts and in turn harder to model. Lastly, during the design phase of this work some other generation methods were explored, among which Triplet Variational Auto-Encoders (TVAE) and TableGAN [102]. During this phase CTGAN showed by far the most potential in terms of quality of the generated samples and practicality, as it allows for control of many aspects of the training process, as well as provide the model designer with a straightforward method to obtain synthetic samples.

8.3.2. Training

After taking the pre-processing steps described in section 8.1 we obtain a suitable training dataset for the CTGAN, but there are still some considerations to be made. The CTGAN framework requires the model designer to determine a set of hyperparameter values to guide the training process. The CTGAN framework defines the following list of hyperparameters [102]:

- **Embedding dimension**: an integer value describing the embedded size of the sample passed to the generator. The default value is 128;
- **Generator dimension**: an integer value depicting the size of the output samples for each residual layer used in the generator. A residual layer is created for each sample. The default value is (256, 256);
- **Discriminator dimension**: an integer value depicting the size of the output samples for each linear layer used in the discriminator. A linear layer is created for each sample. A residual layer is created for each sample.
- **Generator learning rate**: a floating point value representing the learning rate of the generator;
- **Generator decay**: a floating point value representing the weight of the decay for the Adam Optimizer [105] of the generator;
- **Discriminator learning rate**: a floating point value representing the learning rate of the discriminator;
- **Discriminator decay**: a floating point value representing the weight of the decay for the Adam Optimizer of the discriminator;
- **Batch size**: an integer value expressing the amount of training data samples to process each step;

- **Discriminator steps**: an integer value representing the number of discriminator steps to take for each generator step. The default value for the CTGAN implementation is 1, which we follow;
- **Log frequency**: a boolean value determining whether to use log frequency sampling for categorical features; and
- **Epochs**: an integer value describing the number of training epochs.

8.3.3. Parameter tuning

As mentioned in chapter 3.1, many relationships and patterns may exist in the set of fraudulent transactions. Since real-world data, especially tabular data, can become very complex, we want our model to be able to accurately represent this. Creating a suitable set of features in our training set helps to capture such complex behavior, but also comes at the risk of our trained model becoming overly complex and overfitted to the training dataset. Ideally, we want to properly model the complexity found in this data, while still creating a model that generalizes well to new samples and is widely applicable across the field. For this reason, it is interesting to explore different configurations of our generator and evaluate their impact on the quality of the generated data. We will only consider hyperparameters associated with the generator and not with the discriminator. The reason for this is that tuning the parameters of the generator impacts the quality of the generated samples, while the parameters of the discriminator mostly impact the rate at which the network converges.

Generator dimension

The generator dimension hyperparameter represents the number of residual layers in the generator. Increasing the number of these layers allows the generator to capture more complex relationships, but comes at the risk of overfitting [44]. In our use-case, it is interesting to explore this hyperparameter to find a configuration that is able to capture most relevant relationships and patterns found in the training data, whilst remaining generally applicable and not overly specific to the samples found in the training dataset.

Generator learning rate

The generator learning rate controls the rate at which the generator updates its weights as a response to the loss associated with the current configuration. A low learning rate means the generator will take smaller and more conservative steps towards creating increasingly realistic synthetic samples. Tuning the learning rate introduces the trade-off between training speed and generator loss, as well as the trade-off between exploration and exploitation. Higher learning rates will favour exploration in terms of how differently constructed samples are able to fool the discriminator, whereas lower learning rates will favour fine-turning samples to fit the training set distribution.

Generator weight decay

The generator weight decay helps control the degree in which the generator is able to generalize. We do not want our generator to become overly complex, in other words to overfit to the training data. The weight decay parameter allows us to use a wide range of features to capture the complexity found in the training data by limiting the growth of the generator weights to prevent the model overfitting to the training data. A well-tuned value for the weight decay allows for the model to capture complex behavior, while preventing the model from getting too complex.

Both the relationships captured by the generator as well as the degree of overfitting can be measured by the machine learning utility and is done so in experiment 12.1. Tuning these three hyperparameters, along with the two different sampling methods are the main focus of experiment 12.1, where the aim is to find a CTGAN configuration optimizing the similarity to the real dataset, machine learning utility and degree of privacy preservation.

8.4. Dataset reconstruction

After the training process, we obtain a model that is able to generate samples from a joint distribution that closely resembles the joint distribution of previously encountered fraud cases. However, since the CTGAN is only trained on independent features, post-processing of these samples is still needed to obtain samples that resemble the feature arrays of real-world fraud cases. Even then, after constructing the dataset, there is the need to filter out samples that are not suitable for training. To this end, two reconstruction methods called *rejection sampling* and *transformation sampling* are introduced that follow a logical elimination process. Experiment 12.1 will compare these methods against each other on existing benchmarks, as well as analyse the quality of the generated data.

8.4.1. Sample reconstruction

As mentioned previously, the CTGAN will only learn the joint distribution of the aforementioned independent features. As a result, the generated samples logically also consist of this same subset of the total feature set. Fortunately, we possess all tools necessary for reconstructing the original feature set per transaction, as we can simply re-use the logic that was used to determine the features in the first place.

For every feature in the full feature set, simple logic like the above is enough for reconstruction. Even though the CTGAN is able to generate samples that are of high quality and that capture the relationships and patterns similar to the training dataset, it is not free of small mishaps. For this reason, two sampling methods are integrated in the reconstruction process that both aim to select only the samples that most realistically represent feature sets of real transactions.

8.4.2. Rejection Sampling

Rejection sampling is a straightforward way of selecting realistic samples. Even though the samples produced by the CTGAN are sampled from the joint distribution of all previously seen fraud cases, there are instances in which the set of features does not follow the natural rules imposed on the feature set. The general idea of rejection sampling is that samples containing an ’impossible’ set of feature values are dropped from the total set of generated samples. This way, only samples that represent fraudulent transactions that could be encountered in the real world are selected. The reasoning behind this is that a fraudster can directly influence the values of these features.

Using such relationships between all features, the complete feature set is reconstructed, while all illegal samples are dropped during the reconstruction process. As a result, we end up with samples that still accurately represent a sample drawn from the joint fraud distribution and follow the natural set of rules that also holds for all real-world fraud cases. One caveat of using this method, is that it becomes difficult to accurately model the aggregated features used by the model. As a result, in order to maintain the distribution of the sample, constraints based on aggregated features are exempt from being rejected.

8.4.3. Transformation Sampling

Another fairly straightforward way of reconstructing samples that follow the natural rules of transactions is transformation sampling. Instead of rejecting a sample outright, we transform the feature values based on related feature values.

This way, we enforce the aforementioned relationship between the monetary amounts and timespans. Transformation sampling produces samples that are as realistic as rejection sampling, while not suffering from the samples lost due to increased complexity. On the other hand, transformation sampling has two main downsides: the main downside is that the sample, after having its values changes, no longer follows the exact joint distribution that it was sampled from. As a result, some subtle patterns learned from the real dataset might be lost. Secondly, the transformation changes are more computationally demanding than simply dropping the sample. This presents the model designer with the trade-off between dropping certain realism constraints in favour of optimizing the time complexity. Experiment 12.1 compares both methods in terms of the quality of the generated data, to see how real-world realism compares to adherence to the joint fraud distribution of the samples in terms of quality of the machine learning model trained on this data. On top of this, Experiment 12.1 also compares the degree of privacy that both methods are able to achieve.

9 Method: Adversarial Filtering

As described in section 8, even though the synthetic fraud samples generated with the CTGAN are sampled from a joint distribution similar to real fraud cases that we have seen before, some samples are expected to be more beneficial to the improvement of the generalizability of the supervised model than others. Carlini and Wagner define adversarial examples as samples that are very similar to one of the target classes of the machine learning model, but that are classified incorrectly [16]. Existing research in the field of adversarial machine learning has shown that many machine learning models are sensitive to slight alterations in their input, which often results in miss-classifications and errors [2]. Following the logic introduced in chapter 6, we equate such cases to UUs and consequently false negatives. After all, the goal of a fraudster is to create a fraudulent transaction in such a way that it belongs to one of the blind spots of the FDM and is in turn classified as being a legit transaction with high confidence.

An adversary with the goal of making as many fraudulent transactions as possible while going undetected is to find such alterations to in turn fool the machine learning model into thinking his transactions are legit. This section aims to create a method to construct such adversarial examples to in turn feed these to the FDM to improve its robustness against these adversarial strategies. To this end, a method used to quantify the adversarial prowess of a single transaction is introduced. It aims to adopt the mindset of a fraudster and exploit any available knowledge in order to ”defeat” the existing machine learning model. To this end, both data-driven and human-expert-driven techniques are employed.

9.1. Uniqueness scores

As a starting point, we want to create a score to measure the degree in which transactions are different from the average transaction that we have seen before. The idea behind this is that fraud cases that are very similar to the cases already found in the training set are of little added value to improve the robustness of the model against never-seen-before cases. Cases that are sampled from the joint distribution that represent fraud that is sufficiently unique on the other hand are more interesting towards achieving this goal. Because we are working with tabular data, we will introduce methods to determine a *uniqueness score* for both numerical and categorical features. The two scoring methods together map a list of feature values of length n of a single transaction to a list of size n with values representing the uniqueness of each feature value.

9.1.1. Z-scores for numeric features

To determine how unique a certain value for a numerical feature is compared to the average value found in the training data, we will use the *Z-score*. The Z-score for each numerical feature f_i with value x_{f_i}, average feature value μ_f and feature standard deviation σ_f is calculated as follows:

$$S_{Z_{f_i}} = abs\left(\frac{x_{f_i} - \mu_f}{\sigma_f}\right)$$

In more understandable terms: this score indicates how many standard deviations a certain feature value differs from the average value of this feature. Hence, the higher this score, the more unique this value is as compared to the average score. While this score is applicable to the majority of the features, for some features this value only becomes interesting if it is either higher or lower than the average. As an example, for a feature representing the time since the user has signed up to the time of the transaction, both a value that is lower and higher than the average value of this feature is interesting.

Lastly, no normalization step is needed between features after calculating the Z_scores. Because we calculate the uniqueness expressed as the number of standard deviations from the mean per feature, the resulting score is already normalized.

9.1.2. Inversed relative frequency for categorical features

Unfortunately we cannot calculate a Z-score for a categorical feature. Instead, for each categorical feature we find the inverse relative frequency to be a good indicator of uniqueness. Using this method, feature values that are found less often across the training data get a higher score. More formally, for each categorical feature value x_{f_i} in each sample i with frequency F_x across the whole dataset with a total of m samples we calculate:

$$S_{Z_{f_i}} = \frac{1}{\frac{F_x}{m}}$$

One caveat of using this method is when categorical features have a very high cardinality. An example of this is a feature for the users country, which has a cardinality of roughly 195. As a result, a country that is very rare in the training data will have a significantly higher score than more frequently occurring countries, which leads to a disproportional difference in scores compared to similar features between transactions. To remedy this, we introduce a clipping factor to add a bound to each relative frequency. More formally:

$$S_{Z_{f_i}} = [\frac{F_x}{m}, k]$$

This way, by playing around wit the value k we are able to limit the impact of extreme outliers, while still rewarding transactions for having an unique value for high cardinality features.

9.2. Shapley scores

Using the Shapley values obtained from our FVAM, we determine a score per feature based on how well this feature performs in adversarial examples. As explained in section 4.2, the Shapley value associated with a certain feature value indicates how much a this feature generally pushes the model in the direction of a certain decision. To re-iterate: in the binary classification scenario of the FDM, a large positive Shapley value means the feature pushes the model towards predicting 1 (a fraudulent transaction), while a large negative Shapley value pushes the model towards predicting 0 (a legit transaction). For global Shapley values found from the FVAM on the training set however, as described in 7.2, we find that features with a **positive Shapley value** can be exploited to make the FDM not detect fraudulent transactions. In the same sense, features with a **negative Shapley value** help the model to detect fraudulent transactions. In other words, we follow theorem 2.

To construct our adversarial samples, first we min-max normalize the global Shapley values that we find for the training data. Then, we transform the normalized Shapley values $v_{shap_{f_i}}$ for each feature f_i, which are now numbers in $[0, 1]$, into scores $S_{shap_{f_i}}$ in the following way: $S_{shap_{f_i}} = 1 + v_{shap_{f_i}}$. This way, we ensure that features with a positive Shapley value, which we want to reward, get a score higher than 1. Meanwhile, features with a negative Shapley value, which we want to penalize, get a score below 1. These scores indicate per feature the degree in which a specific value is able to ”beat” the model. In other words, these values now represent a weight conforming to the expected adversarial prowess of each feature value in a data-driven way. We call this score the *Shapley score*. Because the Shapley score favors samples that share characteristics with false negatives we expect this score to help us discover UUs.

9.3. Perceptibility scores

On top of using the data-driven method based on Shapley values, we want make use of the vast amount of human expert knowledge found at financial institutions, in our specific case at bunq, as the result of decades of fraud prevention. As the damages caused by financial fraud increase, so does the number of experts working in the field of compliance, with the goal of detecting, analyzing and countering fraudulent or otherwise suspicious payments. As an adversary, one wants to find a way to make transactions that go undetected by these experts. Because most of the transactions in the training dataset of the FDM are labeled by these experts, their approach and mindset has a direct influence on the classifications made by the FDM. On top of this, as we learned from section 4.2, humans are intuitively good at recognizing UUs. For these reasons, we create another score to quantify the degree of likeliness of a transaction is to go undetected per feature based on human expert intuition. We call this score the *perceptibility score*.

The approach towards finding these scores follows the same way of thinking as work done by Ballet et al. [2]. Their approach applies to tabular data and is focused on creating imperceptible adversarial examples. Perceptibility in the context of tabular data requires an extra thinking step compared to image data, which is the field where the majority of the research towards adversarial examples has been done. As opposed to images, tabular feature values are not as interchangeable a pixel values. A natural way to see if an image is an adversarial example is for a human to check the image and its target class. For tabular data, this ”perception check” is more complicated, as tabular data is much less readable and in most cases expert knowledge is required. When comparing two tabular instances, for instance transactions, the expert is most likely to focus on only a subset of the features. These features are the ones that the expert deems important for the distinction between target classes, in our case fraudulent or legit. As a result, the expert is more likely to detect alterations made to this subset of features as these values will be investigated more thoroughly.

From the standpoint of the adversary, one should avoid making modifications to this subset of features. In contrast, the adversary wants to make more significant modifications to the subset of features that are of less importance to the human expert. This mindset has the goal of creating adversarial examples that are altered enough to fool the machine learning model, as an extension of being able to fool a human expert. To achieve this, we create scores that penalize heavy modifications to features with a high human-expert-importance, while rewarding modifications to features with low human-expert-importance. To reduce the bias of one single human expert, the rankings of all features were collected from a total of **8** human experts.

9.3.1. Creating perceptibility scores

Firstly, the average of all human expert rankings is taken for each feature. As mentioned, each expert has his or her own bias with regards to how important each feature is deemed. By taking the average of multiple different experts importance ranking, this bias is mitigated. Before the average is taken, the importance rankings are min-max normalized. The higher the normalized average importance ranking is, the higher the perceptibility of changes to these features is. For this reason, the relationship between expert importance rankings and the resulting scores is inverse linear. We find the feature perceptibility scores $S_{percept_{f_i}}$ in the following way: $S_{percept_{f_i}} = 1/(2 * \tilde{\mu}_f)$, where $\tilde{\mu}_f$ is the normalized mean of the human expert importance rankings. To illustrate, any normalized average expert importance values below 0.5 would get a perceptibility score above 1 and vice-versa for importance values above 0.5. This way, the more important a feature is deemed by human experts to determine whether a transaction is fraud or not, the more changes to values of such features get penalized. Since this score penalizes samples that are **known** to the human experts, while promoting samples that are likely to be **unknown**, we expect this method to promote the discovery of UUs and in turn false negatives.

9.4. Adversarial success scores

To summarize, we started off with an array of values for each transaction describing how unique the feature values of this transaction are. Then, we determine a score for every value based on its corresponding Shapley value describing on how well it is expected to be able to fool the machine learning model. Lastly, we determine another score based on their perceptibility to human experts, which is also aimed at fooling the FDM. Combining these scores, we then calculate what we call the *adversarial success scores* for each feature f_i in the following way:

$$S_{success_{f_i}} = S_{Z_{f_i}} * S_{shap_{f_i}} * S_{percept f_i}$$

For any transaction t with features $f_0, ..., f_i, ..., f_n$ in the feature set with length n we then obtain the aggregated adversarial success score simply by taking the sum of all adversarial success scores in the array:

$$S_t = \sum_{i=1}^{n} S_{success_{f_i}}$$

The resulting value aims to give an indication of how well the transaction will serve as an adversarial example to the current FDM. This chapter will conclude by introducing several ways of selecting the top adversarial examples from a collection of samples, as well as present a way to explain **why** these samples are a good adversarial example.

9.4.1. Other parameters

To allow for exploration in the way adversarial samples are selected, we use a weight per score to determine its impact on the aggregated success score. Furthermore, when determining the uniqueness scores, because the distinction is made between numerical and categorical values, for both calculation methods a weight is introduced to configure the impact of numerical and categorical feature values respectively. We also add a weight to both the Shapley score and the perceptibility scores. Lastly, as mentioned before, the parameter k is used to limit the impact that high cardinality categorical features might have on the success score. Experiment 13.2.2 explores the effect of the scores through their associated weights on the adversarial strength of the produced synthetic samples. In this experiment, the top adversarial examples are ran through the existing FDM to evaluate the degree in which the samples are able to go undetected by the FDM. In other words, how well these adversarial synthetic fraudulent samples represent UUs of the FDM through the rate of false negative classifications.

9.4.2. From unknown unknowns to known knowns

As mentioned in chapter 6, one of the goals of this filtering method is to create a method that turns UUs into unknown knowns. In other words, to teach the model to recognize previously unknown categories fraudulent transactions. This approach has been thoroughly explained in the sections above. In an attempt to turn unknown knows into known knows, which means translating this newly acquired knowledge of the model to something that we as human experts can understand, an explanation method was created using the adversarial success scores. Since we expect our adversarial success scores to be a representation of how likely a sample is expected to be a UU to the FDM, we can leverage the adversarial success scores of the individual feature values $S_{success_{f_i}}$ of the transactions with the highest overall success scores S_t. By listing the names of the features with the highest success scores per transaction, an expert with sufficient domain-specific knowledge might be able to extract patterns in these transactions. To illustrate, we have selected the two highest scoring transactions out of a set of **real** fraudulent samples. This means that we know for a fact that these transactions are fraudulent. This explanation method shows that there is a clear pattern within features of the transactions. As we can see, this method is able to provide meaningful insights for fraud analysts learn about categories of fraudulent transactions that are, according to our scoring system, likely to be UUs.

10 Method: False Positive Mitigation

Using the approaches described in method chapter 8 and 9 we aim to create adversarial synthetic fraudulent samples that help the FDM generalize better by allowing the FDM to learn more information about UUs. However, for this method to be able to favor the discovery of false negatives, it has to consider any other classification type to be of the same category. This consideration is made during the re-labeling process of the FVAM and has been explained in more detail in section 7. For this reason, we expect that the proposed method will come at the cost of an increased number of false positives. Based on what the current goals of the financial institution are, one might seek to reduce either of these. Generally, as mentioned in section 3.1, false negatives are more damaging to the institution and being able to detect otherwise missed fraudulent transactions takes precedence over the losses incurred by an increase in false positives. Nevertheless, false positives in transaction monitoring also cause damages to the company and thus a remedy for the increase in false positives is explored in this section. This approach comes in the form of two relatively simple systems: a system that automatically filters repeated transaction monitoring hits generated by the machine learning model of which we know are not actually fraudulent and another system that aims to help the machine learning model distinguish between different types of fraud.

11 Method: Evaluation Metrics

Defining a proper and suitable set of metrics is essential for evaluating any system. Even more so for the pipeline created in this work, considering the fact that there are various data types and characteristics associated with the techniques that are introduced. Not only do we want to generate synthetic fraud samples that are of sufficient quality, one also requires these samples to guarantee a certain degree of privacy. Furthermore, as the goal of this work is to create robust and high-performing machine learning models, various metrics need to be defined to test such trained models. This section will introduce, motivate and explain the metrics used to evaluate each step of the proposed system.

11.1. Data quality metrics

A multitude of evaluation metrics for the quality of synthetic data exist. The *synthetic data vault* (SDV), an organisation that focuses on the generation of high quality synthetic data, offer a variety of evaluation methods to test the quality of the generated data. These metrics compare the generated samples to the real samples and generally express the quality in terms of its statistical properties. We consider the following metrics defined by the SDV to evaluate how well the generated synthetic data resembles the training dataset [77]:

- **LogisticRegression Detection**: A logistic regression classifier is built to try and tell the real data apart from the synthetic data. The value of this metric is 1 minus the average ROC AUC score of the classifier. Ideally we want to maximize this value, as this means that the ROC AUC is close to 0 and thus that classifier is not able to distinguish between the two datasets.
- **SVC Detection**: Similar to the previous metric, but using a support vector clustering (SVC) classifier.
- **GaussianMixture Log Likelihood**: A statistical metric that tries to fit multiple Gaussian Mixture models to the real data and then computes the average log likelihood of the synthetic data on these models. We aim to maximize this metric, as the higher the log likelihood of the synthetic data to the Gaussian mixture models is, the more similar the distributions of the datasets are.
- **Chi-Squared**: This metric compares the distributions of all corresponding discrete columns of both datasets. It computes the average value of the probability of each two columns having been sampled from the same distribution. For this reason, we aim to maximize this metric.
- **Inverted Kolmogorov-Smirnov D statistic**: Similar to the chi-squared metric, this metric compares the cumulative distribution functions (CDF) of the continuous columns. The value of this metric is 1 minus the average normalized maximum distance of the CDFs of each continuous columns of the real and synthetic data. For this reason, we aim to maximise this metric as well.
- **Continuous Kullback–Leibler Divergence**: This metric computes the entropy between each corresponding continuous column of the real and synthetic datasets. Then, it is normalized in the following way: $\frac{1}{1+KLD}$. We again aim to maximize this metric, as a high KLD indicates a higher degree of seemingly random differences between the dataset.

- **Discrete Kullback–Leibler Divergence**: This metric is the same as the metric above, but for the discrete columns.

Any of these metrics implemented and used in this work make use of the python library provided by the synthetic data vault [77].

11.2. Machine learning utility metrics

In the end, the goal of generating synthetic data is to improve the performance of our supervised model. Consequently, there is the need to evaluate and compare different synthetic samples in terms of their effect on the performance of a model trained on them. Testing the model on a never-seen-before test dataset, the metrics that we are most interested in are the F1 score, ROC AUC, the number of false positives and the number of false negatives. The F1 and ROC AUC scores give a general idea of the performance of the model, while the false positives and false negatives give an indication of how well the model generalizes with respect to the different kinds of errors made by the model. The following metrics are considered in this work, when the aim is to determine the machine learning utility obtained by differently trained models:

In the field of (supervised) machine learning, there exist a wide variety of metrics to evaluate the performance of a machine learning model. This section will consider the most widely used ones and relate their implications to the contributions presented in this work. Because we work in the context of binary classification, only metrics related to binary classification are considered. We consider the following metrics [94].

- **Number of false positives (fp):** the number of transactions that the model has classified as fraudulent, but are in fact legit. Transactions correctly classified as fraud are called true positives (tp);

- **Number of false negatives (fn):** the number of transactions that the model has classified as legit, but are in fact fraudulent. Transactions correctly classified as legit are called true negatives (tn);

- **Confusion matrix**: a (2, 2) matrix which that provides insight into the correctly classified transactions, as well as the different types of aforementioned errors;

- **Accuracy:** describes the overall effectiveness of the model. In essence, accuracy is the rate at which the model makes correct predictions out of the total set of predictions. While the accuracy provides a good overall idea of the performance of the model, it does not tell the complete story. The accuracy is calculated as follows: $\frac{tp+tn}{tp+fn+fp+tn}$;

- **Precision:** the ratio at which the model predicts transactions to be fraudulent transactions that turn out to be actually fraudulent. The precision metric helps us understand how trustworthy a prediction of a transaction being fraudulent is, but fails to represent how many fraudulent transactions were missed by the model. In other words, it does not provide any insights on false negatives. The precision is calculated as follows: $precision = \frac{tp}{(tp+fp)}$;

- **Recall:** the ratio at which the model has made correct predictions, compared to the total number of correct predictions that could have been made. As opposed to the precision, recall helps us understand the degree in which fraudulent transaction have flown under the radar. In our context, it indicates how well the model is able to detect multiple kinds of fraudulent transactions. The recall is calculated as follows: $recall = \frac{tp}{(tp+fn)}$;

- **F1 score:** the f1 score is calculated as the harmonic mean of the precision and the recall of the model and gives a good understanding of the general performance of the model. The F1 score falls short due to the fact that it gives equal importance to the precision and recall while, as we have seen earlier in this report, different mistakes incur different costs. This is not accurately represented by the F1 score. It is calculated as follows: $F1 = \frac{tp}{tp+\frac{1}{2}(fp+fn)}$;

- **F2 score:** similar to the F1 score, the F2 score is calculated using the precision and recall, but gives a higher importance to the recall. This makes the F2 score more suitable in scenarios where false negatives are very costly compared to false positives. In our context, classifying as many fraudulent transactions is more important than maximizing the number of correctly classified transactions. The F2 score is calculated as follows: $\frac{5tp}{5tp+4fn+fp}$; and

- **ROC AUC:** ROC AUC stands for the *Area Under the Curve* of the *Reception Operating Characteristics* curve. In essence, this metric helps us understand how well a model performs at distinguishing between fraudulent and non-fraudulent transactions. The ROC AUC is calculated as follows: $\frac{1}{2}(\frac{tp}{tp+fn}+\frac{tn}{tn+fp})$.

Many of these metrics are used in experiments 12.1 and 12.4 to compare the quality of multiple models trained on different datasets.

11.3. Privacy evaluation metrics

As mentioned briefly in section 5.3.2, the degree of privacy that a synthetic dataset guarantees can be measured in the distance to the original dataset. Unlike the aforementioned metrics, we do not compare the statistical properties or the effects on the machine learning utility, but rather the absolute distance to the original dataset. While this metric is widely applied in the field, it falls short at providing insights on whether smart adversaries are still able to determine sensitive attributes about individuals using the synthetic data. For this reason, we employ one more type of privacy metrics to evaluate a different angle of privacy based on adversarial attacks.

11.3.1. Distance between records

The first and most intuitive metric is the *distance to the closest record* (DCR). This metric is based on the euclidean distance between the records found in the real and synthetic datasets. It indicates the minimum euclidean distance between the two records, one from each dataset. If we find a DCR of 0, it means that the synthetic dataset leaks real information, as it contains a record from the original dataset [76]. We aim to maximize this value, as a higher value indicates that, generally speaking, the values found in the synthesized records bear very little resemblance to any real-world records. Distance-based metrics are a commonly used tool in the field of data privacy protection.

While DCR gives us a decent indication of how close the synthetic data is to the original dataset, it does however have one major shortcoming in our context: due to the random nature involved in the generation of the samples, no matter the technique used, the possibility of a sample that happens to be very similar to a real sample always exists. In this regard, DCR only indicates how well the currently evaluated dataset retains single-record privacy and if the synthetic data is leaking real information, but falls short on explaining the degree of privacy preservation of the evaluated generation method.

To this end, not only do we measure the distance to the closest record, but also the average euclidean distance and the standard deviation between all pairwise distances. Here, we adapt the same mindset as in the work of Park et al. [76]. These authors remark that in order for the synthetic data to guarantee a satisfactory degree of privacy that the average distance is large, while the standard deviation is small. A large average distance indicates that in general, the records in the synthetic dataset provide a good degree of privacy, as these records are not similar to real records. A low standard deviation indicates here that none of the synthetic records are close to the real data. A large standard deviation would indicate that some or multiple pairs between the synthetic and real data exist that are very close.

11.3.2. Adversarial attacks

The aforementioned distance metrics fail to capture the more intricate hidden relationships that might be present between the real and synthetic data. As explained in section 5.3.2, the synthetic data might not be close to any real record, but a smart adversary might still be able to extract sensitive information based on relationships between attribute values. When the data is made public, either through data sharing for research purposes or through a data leak, it is of great importance that no adversary is able to learn such sensitive information and relationships.

This poses the question: given an adversary with access to the synthetic data, can the adversary use a selection of key attributes of the synthetic data to predict the sensitive attributes in the real data? To put this issue to the test for the synthetic data generated in this work, we introduce one last metric: the accuracy of an adversarial attacker model to find sensitive attributes of the real dataset. We train an adversarial attacker on a set of *key attributes* of the synthetic dataset to in turn predict some *sensitive attribute* of the real dataset. The resulting metric is the accuracy of the predictions of this model of the sensitive attributes of the real dataset. This attack is an example of a background knowledge attack and assumes the attacker has access to some external knowledge about the individual in question. In essence, we are measuring the degree of privacy that the synthetic data ensures with respect to these sensitive columns. The first step in calculating this metric is to make a selection of sensitive and key attributes.

Attribute selection
To make a strong selection of the sensitive and key attributes used in the adversarial attack, we need to adopt both the mindset of the victim and the adversary of such an attack. To define sensitive attributes, we ask ourselves the following questions:

- What are characteristics of a person that he or she has no control over?
- What are characteristics of a person that are widely considered to be sensitive from a societal point of view?
- What characteristics of a person would be considered, by the average person, to be unrelated enough to being a fraudster, so that a decision made solely based on this feature would be considered an over-generalization?

Based on these requirements, out of our entire collection of use-case features two features stand out as particularly sensitive:

- **Nationality:** the nationality of the person in question. Right of the bat, this feature clearly checks all of the aforementioned boxes. For an adversary to be able to deduct the nationality of a user would be a significant privacy breach. Furthermore, in the context of fraud, basing the decision of the user being a fraudster solely on this feature would from a societal standpoint be considered as racism. Consider the following fictional example: assume there exists a significant difference in the fraud rate of the transactions made by people from Yemeni descent. Even though this relationship is valuable in the big picture, certain supervised models might assign too much weight to this one relation. If this leads to an unfair classification process for anyone of Yemeni descent, this distinction is definitely problematic. Furthermore, as discussed earlier, false positive fraudulent classifications have various very serious consequences. Therefore, this feature is considered to be highly sensitive.
- **Gender:** the gender of the person in question. Again, this feature ticks all boxes of what we consider to be a sensitive feature. Especially in the current day and age, there exists a significant societal sensitivity surrounding one's gender. Similar the previously mentioned feature, assigning an unfair amount of weight to this feature along with the slew of consequences would be considered to be sexist. For an adversary to deduce this feature and, in turn, make such assumptions would be problematic. As such, this feature is considered to be highly sensitive as well.

Now that we have defined the sensitive attributes, it is important to select a sensible collection of key attributes. From the point of view of an adversary, one would select as many features as possible from the total feature set to base their decision on. Furthermore, one would select only the features that are deemed to be related to the sensitive feature in question. Features unrelated to the sensitive feature would only serve to throw off the classification process. For these reasons, we carefully select the following features to be the key attributes for both sensitive attributes:

- **Nationality:** several features are associated with both the origin of the transaction, as well as the user who has made the transaction. Most of these features are obvious choices, as they are in some way associated with the country of origin of either the user or the transaction. Furthermore, some features describe the nature of the transaction, which are relevant since some countries or users tend to gravitate more towards specific forms of payment, especially international users.

- **Gender:** contrasting the previous feature, very few features are associated with the gender of the user. Still, one could deduct a slew of information based on the social implications of certain features. Many society-focused features have been selected, as there potentially exist many (hidden) relationships between these features and the users gender. For example: in countries where women have very little rights, the gender of the people making transactions will be heavily skewed towards male. Furthermore, the activity associated with the transaction potentially also holds discriminatory information.

Performing an adversarial attack

The adversarial attack is performed using a privacy attacker model based on the *Causative Attack Protection* (CAP) algorithm. The idea behind this algorithm is that an adversary is able to freely inject information into the training data of a supervised machine learning model, with the goal of forcing the model to make wrong classifications [66]. With this knowledge about the poisonous data now present in the training data, the adversary can then in turn make assumptions about the output of the model that the model designer cannot. Adapting this mindset, the algorithm can be adapted to compare datasets and provide the model designer with a means of evaluating the privacy guarantees of one dataset with respect to the other.

In the context of the disclosure of sensitive attributes, the CAP attacker assumes the adversary has knowledge of a set of key attributes, as described in more detail earlier. Using these key attributes, the attacker aims to learn the value of a sensitive attribute. The CAP model allows the model designer to measure the risk of disclosure of an individual's real sensitive target attribute value, in the scenario where the adversary has access to a synthetic dataset [51].

More formally, assume we have a training dataset with n records. For any two records i and j (in this context interchangeable) $j \in \{1, ..., n\}$ this training dataset, let $K_{o,j}$ be the set of values for the key attributes and $T_{o,j}$ the value of the sensitive attribute of record j. In the same fashion, let $K_{s,j}$ and $T_{s,j}$ be the counterparts of the synthetic dataset. The CAP score for the synthetic dataset then becomes [51]:

$$CAP_{s,j} := P_s(To, j|K_{0,j}) = \frac{\sum_{i=1}^{n}[T_{o,i} = T_{o,j} \wedge K_{o,i} = K_{o,j}]}{\sum_{i=1}^{n}[K_{s,i} = K_{0,j}]}$$

The idea behind this equation is that an adversary will search the synthetic dataset for matches of the key attribute values known by them. This subset is referred to as the *equivalence class* of $K_{o,j}$. Using this subset, the adversary will then try to calculate the distribution of the sensitive target attribute with respect to the known key attribute values. In essence, the equation $CAP_{s,j}$ represents the proportion of the target value of the real data $T_{o,j}$ in the equivalence class of $K_{o,j}$. This way, it provides a measure for the risk of the disclosure of the real sensitive target value represented in record j using the knowledge found in its synthetic counterpart. The CAP score is closely related to the notions of k-anonymity and l-diversity described earlier in section 5.3.2.

We use the SDV implementation of the CAP attacker to perform the attacks on our synthetic datasets [77]. As mentioned, this metric outputs the accuracy of a CAP attacker model to correctly classify the sensitive target attribute. As a result, we aim to **minimize** this metric. The details of these attacks, as well as the found results are described in section 12.2 of the experiments chapter.

12 Experiments and Results

To test the viability of the system proposed in this work we conduct a series of experiments. The experiments are set up in a logical order, working from the ground up: first, different configurations of the CTGAN are tested to find their impact on the overall quality of the synthetic samples produced. This quality is expressed as the similarity to the real fraudulent samples, the machine learning utility and the degree of privacy preservation. Secondly, the adversarial filtering method is put to the test by comparing different configurations of the filtering system and evaluating the effects on the samples in terms of adversarial strength. To test this, we run various generated synthetic datasets through the existing FDM and compare the false negative rates found. Thirdly, we create a synthetic dataset using the optimal configurations of the CTGAN and adversarial filtering systems to produce a set of synthetic adversarial examples. We then train various FDMs on combinations of synthetic adversarial examples, real fraudulent samples and real legit transaction samples to test the quality of these models against baseline models. Lastly, we test two methods with the goal of reducing the increased number of false positives introduced by the previous system.

General experimental setup

All datasets used in the experiments share the same parent dataset. We base our experiments on a subset of this total set which consists of 1567 fraudulent transactions and 303424 legit transactions. We will refer to this training set as S_{train} from now on. This set was carefully chosen for the following reasons:

1. It is a smaller version of the total transaction dataset. This allows us to conduct experiments on a smaller scale in a shorter time. Especially when training the CTGAN or XGB model this saves a significant amount of time; and

2. There is a serious class imbalance in this dataset. The number of legit transactions heavily outweigh the number of fraudulent transactions. As this is one of the main challenges that this work aims to solve, it is crucial for the experiments that this dataset satisfies this property;

Furthermore, since many different configurations of both the CTGAN and adversarial filtering system are used, each experiment will indicate the specific configurations used and their implications. When evaluating the FDMs trained on different trainging sets we make use of the same test dataset. This test set consists of roughly 80000 real fraudulent transactions and roughly 200000 real legit transactions. We will call this test set S_{test}. We use a large number of fraudulent transactions in our test set, as to have a wide variety of different real fraudulent transactions. This allows us to accurately test the robustness of the trained models to never-seen-before fraudulent transactions. S_{test} has been carefully constructed to ensure no overlap between S_{train} and S_{test} exists so that S_{test} is a completely never-seen-before set of transactions to any of the evaluated FDMs. For this reason we can safely make assumptions about the obtained robustness of the trained models.

12.1. Comparing different methods of generating synthetic samples

This experiment aims to find out whether the generated synthetic samples are able to properly convey the relationships and patterns found in real fraudulent samples. To this end, we compare the effects of multiple configurations of the generation process, as well as the two reconstruction and sampling methods introduced in section 8.4 on the quality of the generated synthetic samples. In this section, we consider the quality of the synthetic samples as expressed by the similarity of the samples to the real fraudulent samples, as well as the machine learning utility provided by the synthetic samples. In the next experiment section, we explore the quality in terms of the degree of privacy preservation.

12.1.1. Similarity to the real data

For the purpose of gaining an understanding how closely the generated fraud samples resemble the real fraud cases, we calculate a subset of the metrics introduced in section 11.1. We choose the following metrics:

1. **LogisticRegression Detection** to find out how well a logistic regression classifier can distinguish between synthetic and real samples;

2. **Chi-Squared** to find how similar the distributions of the discrete columns of the synthetic data are to their respective distributions found in the real data. In the same breath, one could say that this value indicates the overall similarity found in the categorical columns;

3. **Inverted Kolmogorov-Smirnov D statistic** to find how similar the distributions of the continuous columns of the synthetic data is to the distributions of their respective columns found in the real data. In the same breath, one could say that this value indicates the overall similarity found in the numerical columns;

Experiment setup

Each CTGAN is trained on the same set of 1567 fraudulent transactions from S_{train}. The experiment is set up as follows:

1. Create a hyperparameter configuration and train a CTGAN on the real fraudulent transaction samples. We limit our exploration to different values for the following hyperparameters: generator learning rate, generator weight decay and generator dimension. The definitions of these hyperparameters, along with the motivation behind their selection is detailed in section 8.3.2. When tuning any hyperparameter, the values of the other hyperparameters are left unchanged as to isolate the effects of a single hyperparameter;

2. Using both rejection and transformation sampling construct two datasets of 1567 synthetic fraudulent samples from the particular CTGAN; and

3. Find the similarity of these datasets to the real fraudulent samples found in S_{train} by computing the aforementioned metrics;

From step 2 and onwards, we perform multiple rounds and take the average value of the evaluated metrics over all rounds as to mitigate the effect of the randomness of the samples taken from the CTGAN as much as possible.

The LogisticRegression Detection score

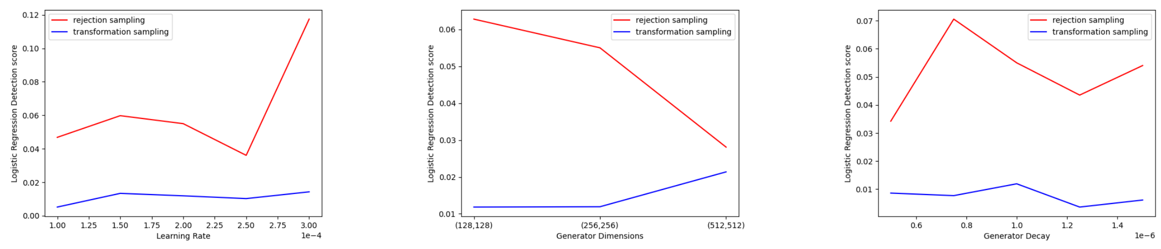


Figure 12.1: LogisticRegression Detection scores for samples obtained from different configurations of the CTGAN

The Chi-Squared score

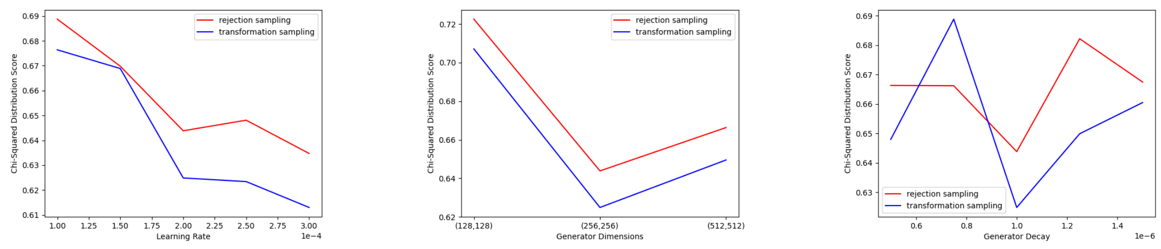


Figure 12.2: Chi-squared scores for samples obtained from different configurations of the CTGAN

The Inverted Kolmogorov-Smirnov D statistic

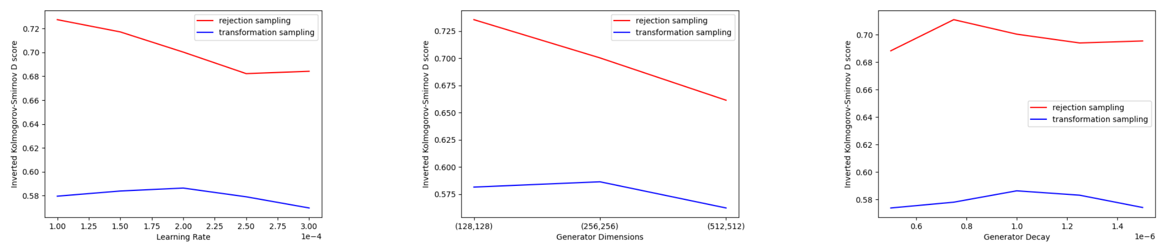


Figure 12.3: Inverted Kolmogorov-Smirnov D scores for samples obtained from different configurations of the CTGAN

12.1.2. Machine learning utility

In order to gain an understanding of the machine learning utility provided by the synthetic samples, we compare multiple samples generated by different configurations of the CTGAN to a baseline of real fraud samples. The idea behind this, is that for real fraud samples the machine learning utility is ”as good as it gets”. By comparing differently configured CTGANs, we aim to find the configuration of hyperparameters that produces synthetic samples that provide the highest machine learning utility.

Setup

Again each CTGAN is trained on the same set of 1567 fraudulent transactions from S_{train}. Each round of this experiment is set up as follows:

1. Create a hyperparameter configuration and train a CTGAN in a similar fashion as the previous round of experiments;
2. Using both rejection and transformation sampling construct two datasets of 1567 synthetic fraudulent samples from the particular CTGAN;

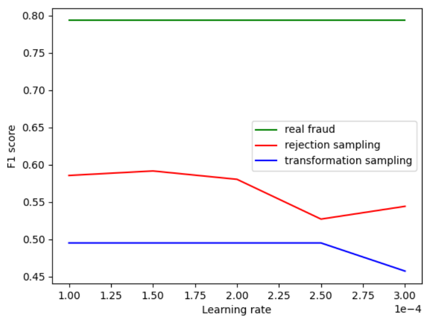
3. Create a training set S_i with this sample and the set of real legit transactions from S_{train}, as to emulate the class imbalance of the real-life setting as much as possible;

4. Train a FDM on S_i;

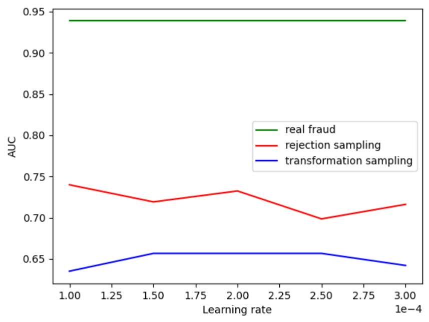
5. Calculate the F1 score and ROC AUC of the model on S_{test}.

From step 2 and onwards, we perform multiple rounds and take the average value of the evaluated metrics over all rounds as to mitigate the effect of the randomness of the samples taken from the CTGAN as much as possible.

Generator learning rates



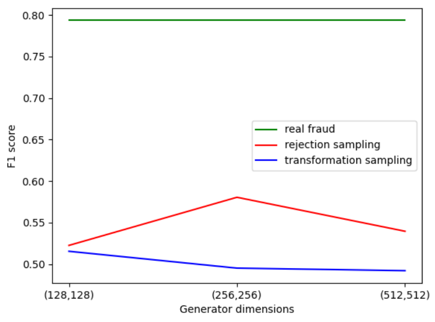
(a) The effect of different learning rates on the F1 score of a supervised model trained on the synthetic samples



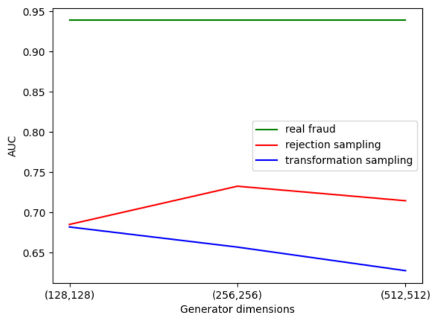
(b) The effect of different learning rates on the ROC AUC score of a supervised model trained on the synthetic samples

Figure 12.4: The machine learning utility metrics for samples obtained from different learning rates of the CTGAN

Generator dimensions



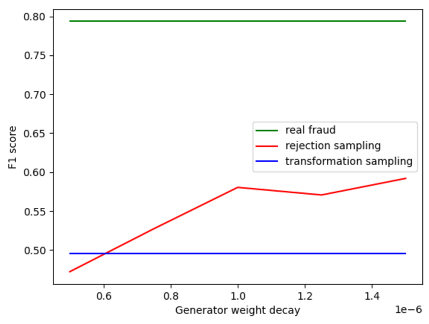
(a) The effect of different generator dimensions on the F1 score of a supervised model trained on the synthetic samples



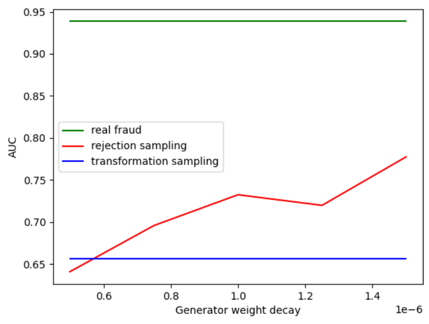
(b) The effect of different generator dimensions on the ROC AUC score of a supervised model trained on the synthetic samples

Figure 12.5: The machine learning utility metrics for samples obtained from different generator dimensions of the CTGAN

Generator weight decay



(a) The effect of different generator decay weights on the F1 score of a supervised model trained on the synthetic samples



(b) The effect of different generator decay weights on the ROC AUC score of a supervised model trained on the synthetic samples

Figure 12.6: The machine learning utility metrics for samples obtained from different generator weight decay values of the CTGAN

12.2. Evaluating the privacy guarantees of the synthetic data

As described in section 5.3.2, synthetic data provides various advantages in terms of data privacy in case of data leaks or for publishing anonymized datasets for data mining purposes. In this section the privacy guarantee of the synthetic data will be evaluated using multiple metrics to test different aspects of privacy. We will test the quality in terms of privacy of the synthetic data by calculating the distance to the closest record, performing an adversarial attack on two sensitive attributes found in our case study dataset and finally analysing the trade-off between privacy-guarantee and machine learning utility.

12.2.1. Setup

Similar to experiment 12.1.2, multiple differently configured CTGANs are trained on the same set of 1567 fraudulent transactions from S_{train}. The experiment is set up as follows:

1. Create a hyperparameter configuration and train a CTGAN in a similar fashion as the previous round of experiments;
2. Take two samples using both rejection and transformation to obtain a sample S_i consisiting of 1567 synthetic fraud cases from the particular CTGAN; and
3. Calculate or determine the values of the aforementioned privacy metrics based on S_i and the real fraudulent transactions from S_{train} as described in method section 11.3;

From step 2 and onwards, we perform multiple rounds and take the average value of the evaluated metrics over all rounds as to mitigate the effect of the randomness of the samples taken from the CTGAN as much as possible.

12.2.2. Distance to the closest record

To calculate the distance to the closest record, a custom nearest-neighbour based function for our use case was implemented as specified in method section 11.3. The tables below show the distance to the closest record, the average record distance and the standard deviation of the record distances for each configuration. The most striking results are indicated in red and green. Red indicates that the value or value combinations are among the least desirable over all the experiments. Green indicates the opposite.

Table 12.1: The distance metrics obtained for multiple generator learning rate values of the CTGAN using rejection sampling

	1-e4	**1.5e-4**	**2e-4**	**2.5e-4**	**3e-4**
To closest record	0.085	0.083	0.090	0.088	0.093
Standard deviation	0.116	0.115	0.120	0.120	0.122
Average	0.435	0.440	0.444	0.446	0.412

Table 12.2: The distance metrics obtained for multiple generator learning rate values of the CTGAN using transformation sampling

	1-e4	**1.5e-4**	**2e-4**	**2.5e-4**	**3e-4**
To closest record	0.130	0.125	0.129	0.115	0.129
Standard deviation	0.166	0.164	0.167	0.161	0.169
Average	0.393	0.414	0.403	0.434	0.394

Table 12.3: The distance metrics obtained for multiple generator dimensions of the CTGAN using rejection sampling

	(128,128)	**(256,256)**	**(512,512)**
To closest record	0.081	0.090	0.097
Standard deviation	0.115	0.120	0.125
Average	0.459	0.444	0.396

Table 12.4: The distance metrics obtained for multiple generator dimensions of the CTGAN using transformation sampling

	(128,128)	**(256,256)**	**(512,512)**
To closest record	0.128	0.129	0.124
Standard deviation	0.167	0.167	0.165
Average	0.398	0.403	0.430

Table 12.5: The distance metrics obtained for multiple generator decay values of the CTGAN using rejection sampling

	5e-7	**7.5e-7**	**1e-6**	**1.25e-6**	**1.5e-6**
To closest record	0.0861	0.0862	0.0907	0.0885	0.0842
Standard deviation	0.116	0.117	0.121	0.119	0.118
Average	0.431	0.420	0.444	0.416	0.457

Table 12.6: The distance metrics obtained for multiple generator decay values of the CTGAN using transformation sampling

	5e-7	**7.5e-7**	**1e-6**	**1.25e-6**	**1.5e-6**
To closest record	0.129	0.125	0.129	0.131	0.109
Standard deviation	0.166	0.165	0.167	0.167	0.160
Average	0.393	0.394	0.403	0.385	0.427

12.2.3. Adversarial attacks

In this experiment we evaluate the success rate of a CAP attacker to figure out the values of sensitive attributes of the real dataset, based on the values found in the synthetic dataset, as detailed in method section 11.3. On top of testing this metric on different configurations of the CTGAN, we test it on 2 different kinds of sensitive attributes: the nationality of the user and the gender of the user.

Nationality

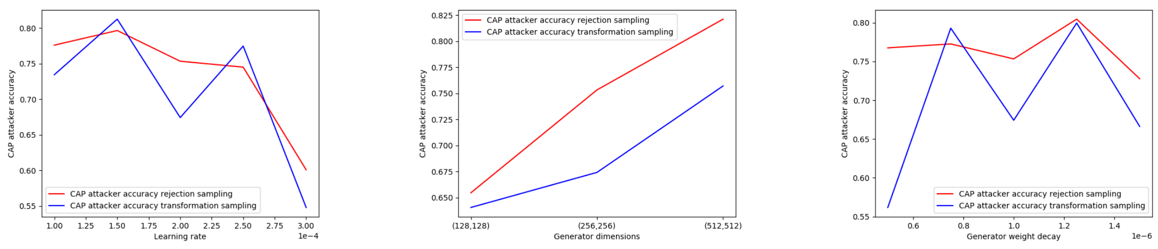


Figure 12.7: Accuracy of the CAP attacker finding the users nationality on samples generated by the three different configuration types of the CTGAN

Gender

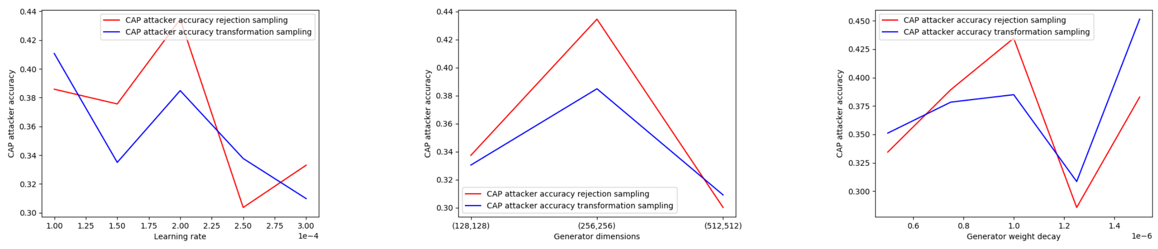
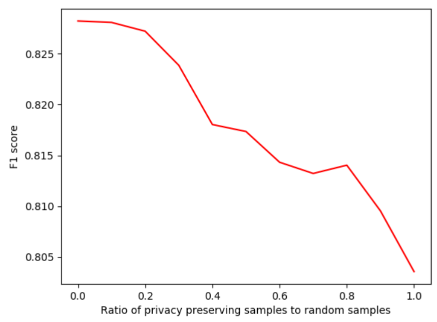


Figure 12.8: Accuracy of the CAP attacker finding the users nationality on samples generated by the three different configuration types of the CTGAN

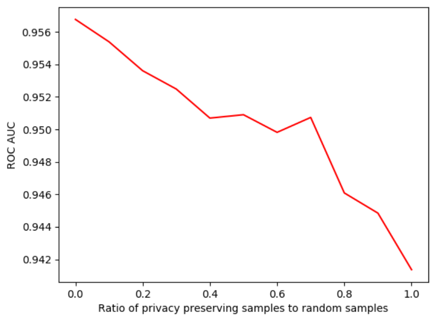
12.2.4. Privacy guarantee vs. machine learning utility

Lastly, we test the trade-off between privacy guarantee and machine learning utilty. This experiment is set up in the following way:

1. We train a CTGAN with default hyperparameter configurations on the fraudulent transactions in S_{train}, similar to the previous experiments;
2. We generate 100 thousand synthetic samples using this CTGAN and rejection sampling;
3. We sort these samples by degree of privacy preservation based on the DCR of and take the top 1567 samples, which we call S_p;
4. We then take 1567 random samples from the same set of 100.000 samples, which we call S_r;
5. We calculate the average DCR of both of these synthetic datasets;
6. We create 10 different datasets of size 1567 consisting of different ratios between the 2 datasets mentioned above, each time taking steps of 0.1;
7. We calculate the F1 score and the ROC AUC as measures of the machine learning utility.



(a) F1 score for different levels of privacy preservation



(b) ROC AUC score for different levels of privacy preservation

Figure 12.9: The machine learning utility metrics for samples obtained from different configurations of the CTGAN

Dataset	**Average DCR**
S_p	0.086
S_r	0.122

Table 12.7: Average distance to the closest record for the two synthetic datasets S_p and S_r

In these figures the privacy guarantee is expressed in terms of the ratio of samples with a high privacy guarantee (high distance to closest record). This means that a ratio of 1 has the highest privacy guarantee, while a ratio of 0 has the lowest privacy guarantee.

12.3. Defeating an existing FDM with adversarially filtered synthetic fraud cases

Whereas the previous rounds of experiments were aimed at gaining a grasp of the quality of the generated synthetic samples, this experiment is aimed at finding the specific samples that the existing FDM has trouble classifying. We operate under the assumption that, after determining that the quality of the samples is sufficiently high, samples taken from a CTGAN trained on only fraudulent transactions count as confirmed fraud cases. Then, using our method for filtering out only the best adversarial samples as described in method section 9, we evaluate the adversarial prowess of these samples by running them through the existing FDM trained on S_{train}, which we will refer to as $FDM_{default}$ from now on, and comparing the classification results to several baselines.

12.3.1. Setup

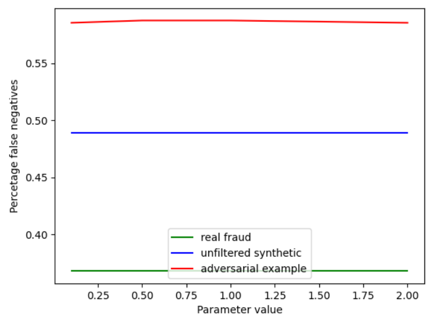
The adversarial examples are compared against two baselines in the form of an equal number of real fraud samples and random synthetic samples. The experiment is set up in the following way:

1. Take a CTGAN with the configuration that maximises the trade-off between machine learning utility and degree of privacy preservation of the synthetic samples produced;
2. Generate 100 thousand synthetic samples. Important to note is the fact that we did **not** generate new samples for each filtering configuration. The random nature of this would influence the results, which means we cannot isolate the effects of a single weight being changed;
3. For each configuration that is tested, select the top 1% best adversarial examples from the dataset of the previous step as found by the adversarial filtering method;
4. Add the 1000 adversarial examples to a set of 195000 real legit transactions to create a test dataset that suffers the same rate of class imbalance as the total set of transaction samples does;

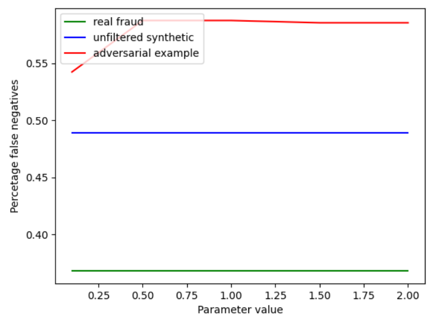
5. Use $FDM_{default}$ to classify this test set. There is no overlap between the test set and the training set of this model, as the fraudulent samples are synthetic and the real legit samples are sampled from a different set than S_{train};

6. To test the adversarial prowess of such a set of adversarial examples we measure the percentage of fraudulent samples that go undetected by the model on each test set. In other words, we measure the false negative rate; and

7. We compare these results to two baselines: the first baseline is a set of real fraudulent samples. These samples were randomly selected from the total set of fraudulent transactions, ensuring no overlap between these samples and the fraudulent samples in S_{train}. This means that the fraudulent samples in the baseline test set have never been seen by $FDM_{default}$. Our second test set consists of random synthetic samples from the set of 100 thousand generated samples.

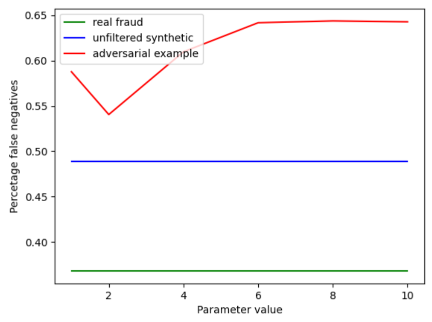


(a) False negative rates for different numerical uniqueness values

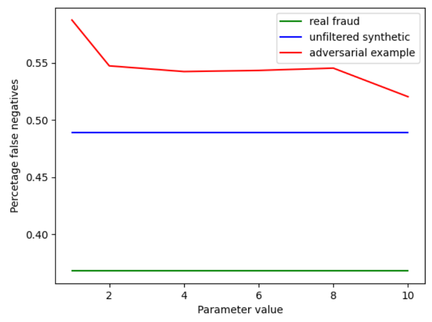


(b) False negative rates different categorical uniqueness values

Figure 12.10: The false negative rates of $FDM_{default}$ on the test dataset as filtered using different numerical and categorical uniqueness weight values

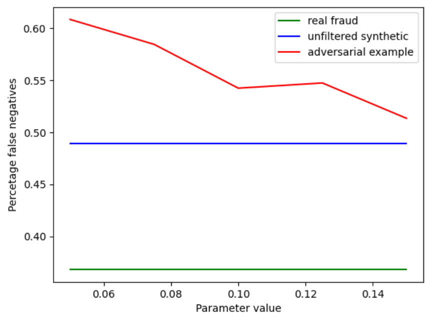


(a) False negative rates for different Shapley weight values



(b) False negative rates different expert perceptibility weight values

Figure 12.11: The false negative rates of $FDM_{default}$ on the test dataset as filtered using different Shapley and expert perceptibility weight values



(a) False negative rates for different categorical cardinality bound values

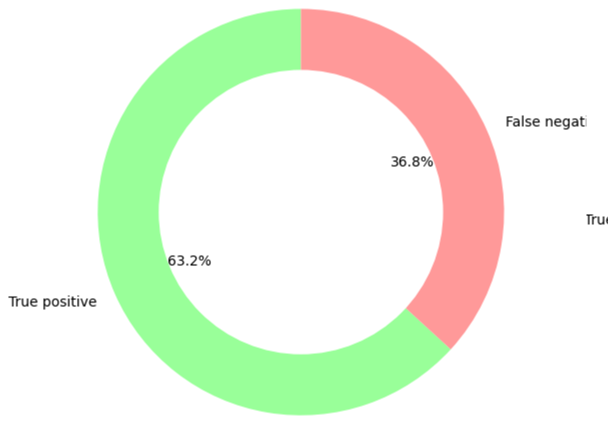
Figure 12.12: The false negative rates of $FDM_{default}$ on the test dataset as filtered using different categorical cardinality bound values

Lastly, to illustrate the adversarial prowess of the different samples we will take the configuration of the adversarial filter that produces the highest number of false negatives and show the error ratios of $FDM_{default}$ in the format of a donut chart. This configuration is shown in table 12.8.

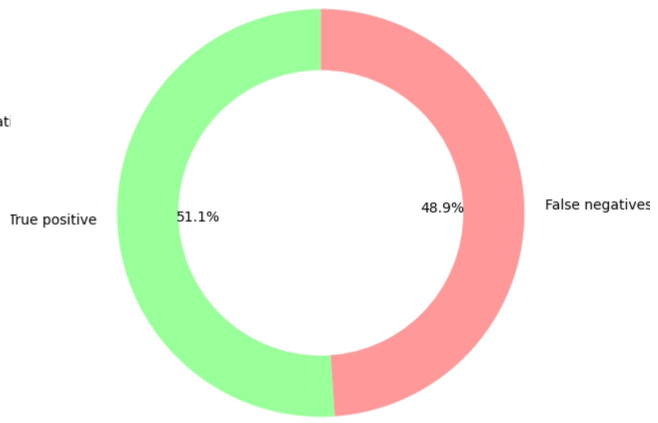
Adversarial filter parameter	**Value**
Numeric uniqueness	1
Categorical uniqueness	1
Shapley weight	6
Perceptibility weight	1
Categorical cardinality bound	0.10

Table 12.8: Adversarial filtering configuration chosen based on the previous findings

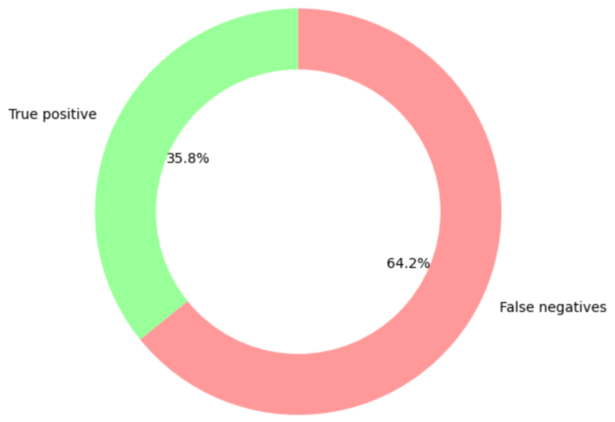
Using the configuration for the adversarial filtering system as shown in table 12.8, we depict the performance of $FDM_{default}$ on the filtered synthetic dataset and the two aforementioned baselines in figures 12.13a to 12.13c. In this figure we find three donut charts, each depicting the total number of fraudulent samples divided into false negatives and true positives as classified by $FDM_{default}$.



(a) Real samples



(b) Random synthetic samples



(c) Adversarially filtered synthetic samples

Figure 12.13: The false negative rates for three test datasets as classified by $FDM_{default}$

12.4. Training a supervised model on synthetically augmented datasets

As the final step to our approach, we compare multiple FDMs trained on real fraud samples, synthetic fraud samples and both. Using the top adversarial examples explored in the previous experiments, we aim for these FDMs to obtain a higher robustness to never-seen-before real fraudulent samples and consequently a higher performance than $FDM_{default}$.

12.4.1. Setup

We again use S_{train} as in experiment 12.1, consisting of 1567 fraudulent and 303424 legit real transactions. This experiment is set up as follows:

1. We use the same 100 thousand synthetic samples generated for the previous experiment;
2. Using the adversarial filtering system with the configurations shown in table 12.8 from experiment 13.2.2, we select the top 783, 1567 and 3134 synthetic fraudulent transactions;
3. We then train a supervised model on combinations of the synthetic datasets mentioned above and all real fraudulent samples at our disposal, along with all real legit samples; and
4. Finally, we compare the results to one another and some important baselines: $FDM_{default}$ which is trained on the real fraudulent samples from S_{train}, a model that trained on only random synthetic samples in terms of fraudulent samples and a model trained on only adversarial examples in terms

of fraudulent samples. All of these training datasets also include the full set of real legit transaction samples.

To summarize, we create the following training datasets:

1. **Dataset A:** 1567 real fraudulent transactions and 303424 legit real transactions;
2. **Dataset B:** 1567 adversarial examples and 303424 legit real transactions;
3. **Dataset C:** 1567 random synthetic samples and 303424 legit real transactions;
4. **Dataset D:** 783 adversarial examples, 1567 real fraudulent transactions and 303424 real legit transactions;
5. **Dataset E:** 783 random synthetic samples, 1567 real fraudulent transactions and 303424 legit real transactions;
6. **Dataset F:** 1567 adversarial examples, 1567 real fraudulent transactions and 303424 real legit transactions;
7. **Dataset G:** 1567 random synthetic samples, 1567 real fraudulent transactions and 303424 real legit transactions;
8. **Dataset H:** 3134 adversarial examples, 1567 real fraudulent transactions and 303424 real legit transactions;
9. **Dataset I:** 3134 random samples, 1567 real fraudulent transactions and 303424 real legit transactions;

As the test set, we again use S_{test} consisting of 80.000 fraudulent and 200.000 legit real transactions. All of these transactions are never seen before by any of the models trained for this experiment. In any of the rounds of experiments done in this section, we use the full amount of real fraudulent transactions, as we want to use any and all information of the underrepresented class at our disposal.

12.4.2. Training models on different samples

Using the training datasets described above, we train 9 respective models:

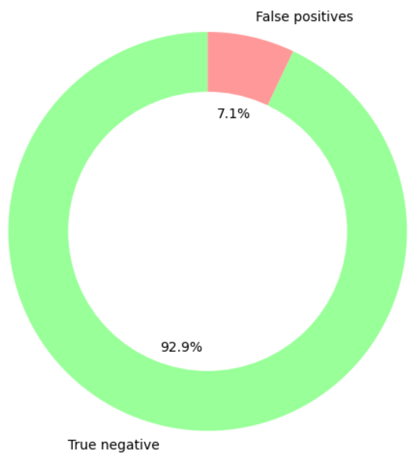
- **Model A:** this model is the exact same as $FDM_{default}$. Hence, this model is considered as the most important baseline for machine learning utility.
- **Model B:** to test the machine learning utility of the base samples produced by the CTGAN, we test the performance of a random sample of 1567 synthetic samples. This allows us to not only compare the machine learning utility of the general synthetic samples compared to the real samples, but also compare the advantages of adversarially filtering the synthetic samples before adding them to the training dataset;
- **Model C:** similar to the previous model, we aim to measure the machine learning utility of the adversarial samples to compare it to the real and random samples; and
- **Models E, G and I:** In order to be able to evaluate the effects adversarially filtering the synthetic samples before adding them to the training dataset, these models are trained on both real fraud and random synthetic samples;
- **Models D, F and H:** To test the advantage of augmenting the training dataset with adversarial samples, models are trained on both real fraud and adversarial examples. Here we are still using all information at our disposal, but use the methods described in 9 to try to increase the model performance.

The results of these training experiments are shown in table 12.9. On top of comparing the performance between using adversarial samples against not using them, we also compare different amounts of adversarial samples when augmenting the training dataset.

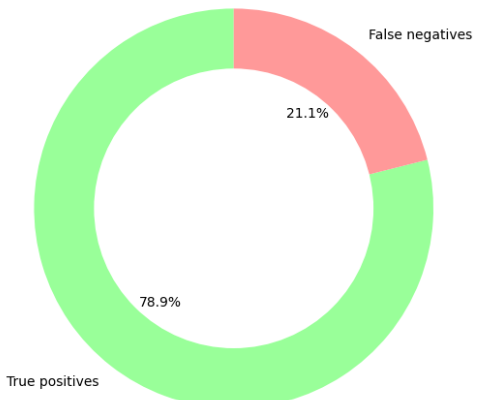
Table 12.9: Multiple metrics indicating the performance of models A to I on S_{test}. Indicated in bold is the best performing model for each metric.

Model	#FP	FP rate	#FN	FN rate	F1	F2	ROC AUC
A	14194	0.0706	16869	0.2109	0.8026	0.8532	0.9283
B	34284	0.1705	37672	0.4709	0.5405	0.6655	0.6787
C	27475	0.1366	40035	0.5004	0.5421	0.6655	0.6912
D	15498	0.0771	12671	0.1584	0.8270	**0.8760**	0.9442
E	11670	0.058	15866	0.1983	0.8233	0.8732	0.9552
F	16254	0.0808	**11777**	**0.1472**	**0.8296**	0.8732	**0.9606**
G	15111	0.0752	14402	0.18	0.8164	0.8557	0.9536
H	14572	0.0725	15416	0.1927	0.8116	0.8513	0.9475
I	**10396**	**0.0517**	18216	0.2277	0.8120	0.8570	0.9467

To further visualize the differences in false positive and false negative ratios, we will show models obtaining the most striking results along with the baselines in the form of donut charts. These charts aim to provide an intuitive and clear perspective of how certain training datasets help the model generalize to never-seen-before samples.

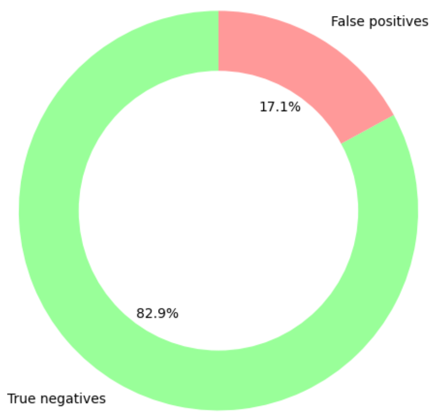


(a) False positive and true negative rates of the FDM

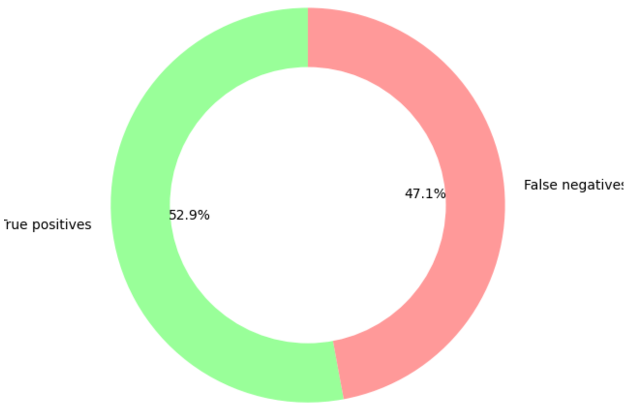


(b) False negative and true positive rates of the FDM

Figure 12.14: The false positive and false negative rates found on the test set classified by **model A**
.

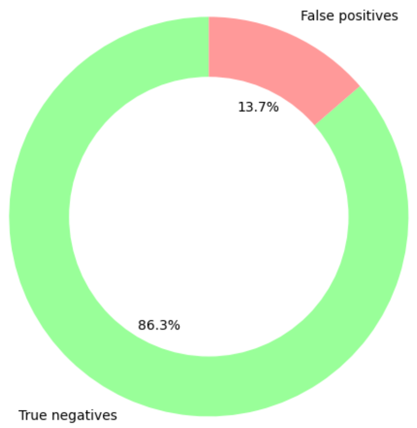


(a) False positive and true negative rates of the FDM

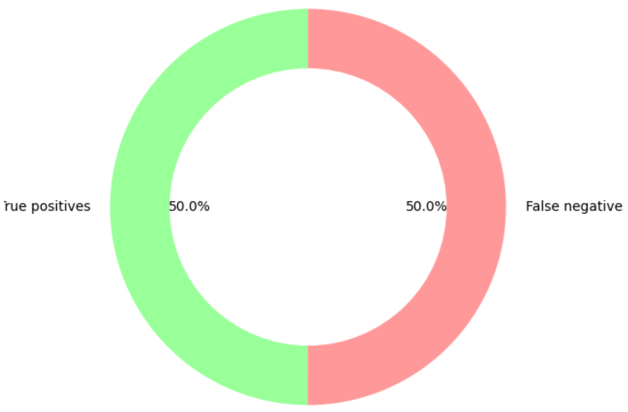


(b) False negative and true positive rates of the FDM

Figure 12.15: The false positive and false negative rates found on the test set classified by **model B**
.

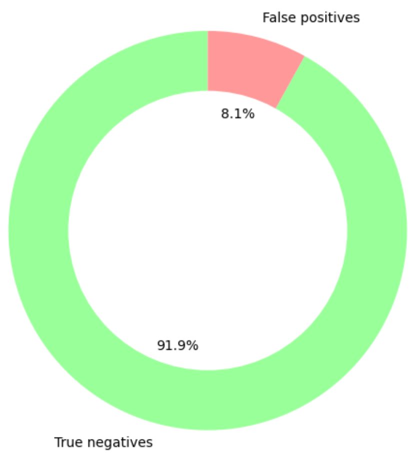


(a) False positive and true negative rates of the FDM

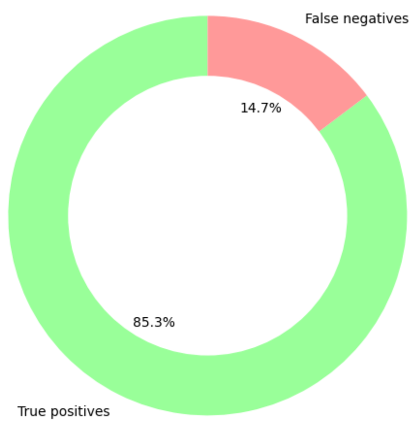


(b) False negative and true positive rates of the FDM

Figure 12.16: The false positive and false negative rates found on the test set classified by **model C**
.

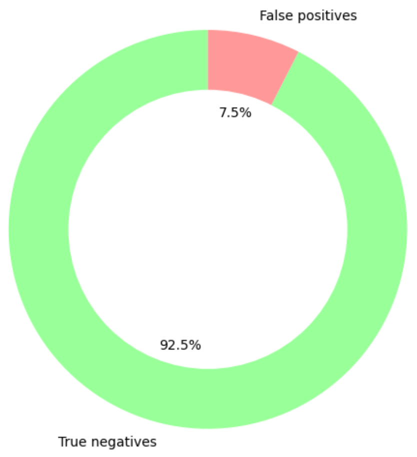


(a) False positive and true negative rates of the FDM

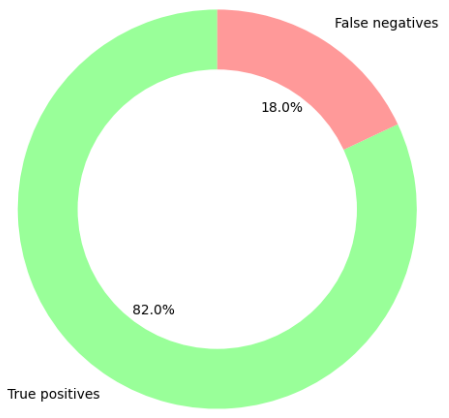


(b) False negative and true positive rates of the FDM

Figure 12.17: The false positive and false negative rates found on the test set classified by **model F**
.

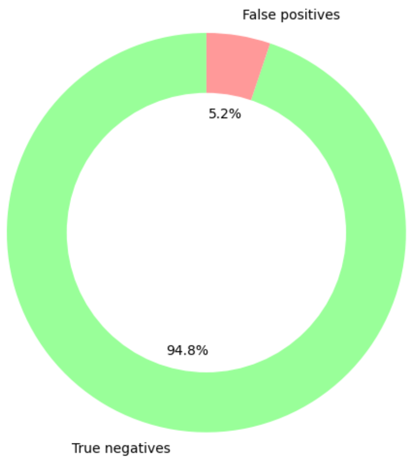


(a) False positive and true negative rates of the FDM

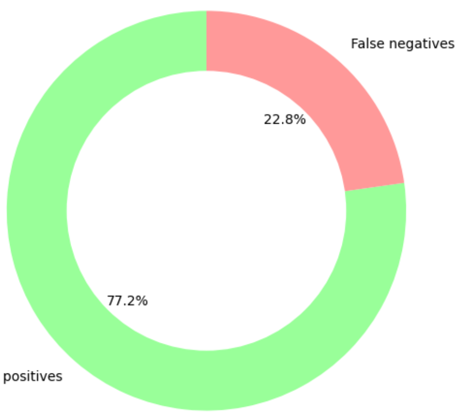


(b) False negative and true positive rates of the FDM

Figure 12.18: The false positive and false negative rates found on the test set classified by **model G**
.



(a) False positive and true negative rates of the FDM



(b) False negative and true positive rates of the FDM

Figure 12.19: The false positive and false negative rates found on the test set classified by **model I**
.

12.4.3. Model performance for different thresholds

The experiments above have all used the optimal threshold as calculated by the H2O framework during the training process, based on the F1 score. In many situations however, it is interesting to evaluate the model performance for different thresholds. We will discuss this in more detail in discussion subsection 13.3.1. Figures 12.20 and 12.21 show the relationship between different threshold values and the number of false negatives and false positives and the F1 score respectively. For this experiment, we select the best performing model that makes use of the adversarially filtered synthetic samples from the previous round of experiments, based on the F1 score. This means we select the model trained on dataset **F**. As baselines, we take model **A** ($FDM_{default}$) and model **C**. We use model A to provide a comparison with the existing scenario to show the benefits of using our proposed method. We use model C to show the advantages of using the adversarial filtering system over random synthetic samples. For both baseline models, we only indicate the performance metrics of when using the optimal threshold, because this accurately represents the default situation.

False positives and false negatives

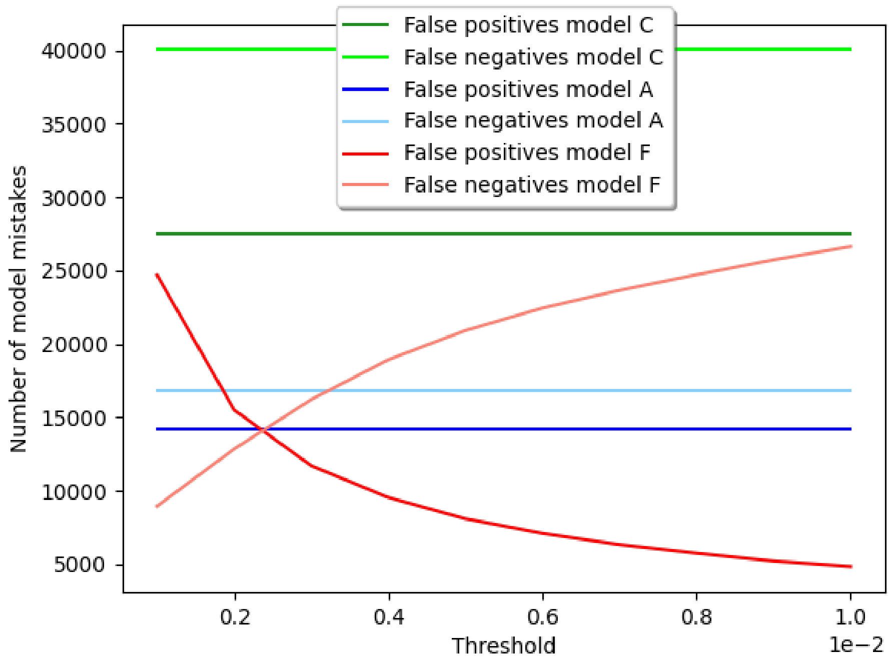


Figure 12.20: The number of false positives and false negatives as found for various classifier thresholds for model F and optimal thresholds for two baseline models.

12.4.4. F1 scores

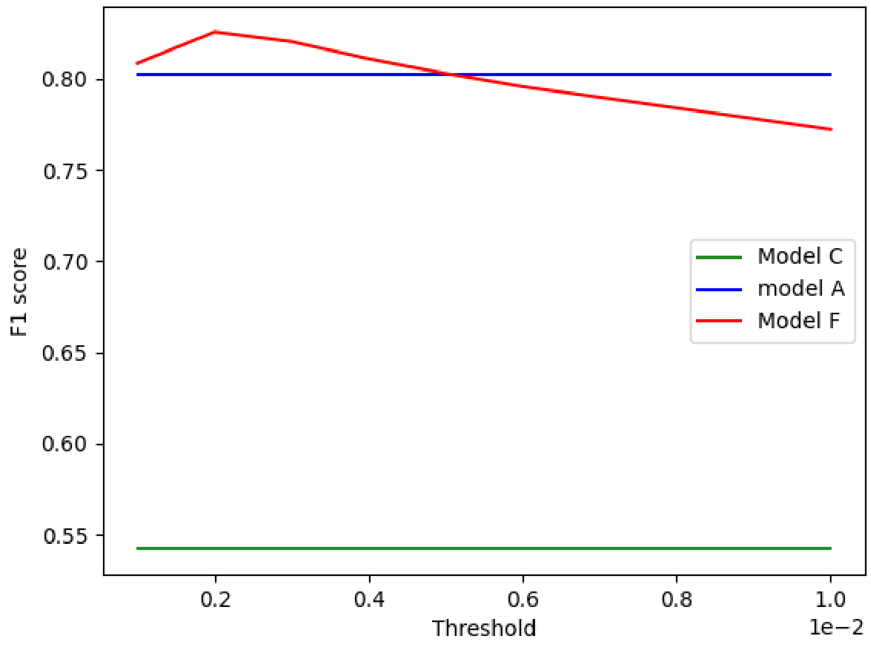


Figure 12.21: The F1 score as found for various classifier thresholds for model F and optimal thresholds for two baseline models.

12.5. Reducing false positives

Since the generalization benefits of our most promising model, model F, come at the price of an increased number of false positives, efforts were made to try control this number. This effort is two-fold: using weights to our training dataset to more accurately represent samples that have shown to be difficult to label as either fraud or legit and implementing a system that filters out repeated false positive hits of the FDM. In this experiment we evaluate both methods.

12.5.1. Using weights to represent edge-cases

In this section we evaluate the approach proposed in method section **??**. The experiment is set up in the following way:

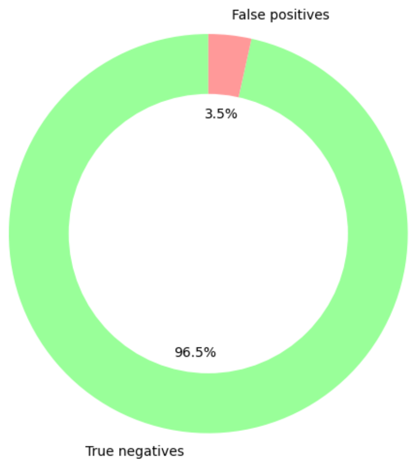
1. We create two datasets **X** and **Y**:

    - Dataset **X** consists of 1567 real transactions where the final conclusion whether the transactions are fraudulent or not is undecided and all of the real legit transactions from S_{train}.
    - Dataset **Y** is an exact copy of dataset **X** with the addition of a weight column to the training dataset. The weights per row are obtained as described in method section **??**.

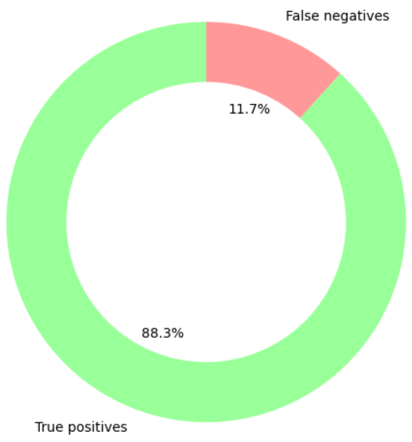
Table 12.10 shows the performance metrics found for two identical models trained on a weighted and unweighted dataset respectively.

Table 12.10: Differences in model performance with and without rule precision weights on undecided cases. Shown in bold is the top performing model per metric.

Model	#FP	FP rate	#FN	FN rate	F1	F2	ROC AUC
X	6987	0.0347	**9361**	**0.117**	**0.896**	**0.909**	**0.978**
Y	**5791**	**0.0288**	11022	0.138	0.891	0.903	0.977

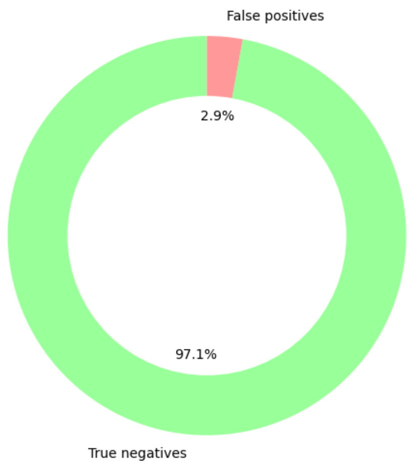


(a) False positive and true negative rates of the FDM

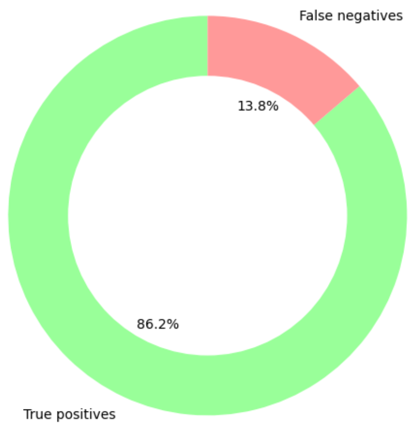


(b) False negative and true positive rates of the FDM

Figure 12.22: The false positive and false negative rates found on the test set of model **X**



(a) False positive and true negative rates of the FDM



(b) False negative and true positive rates of the FDM

Figure 12.23: The false positive and false negative rates found on the test set of model **Y**

13 Discussion

The system created in this work consists of a pipeline of multiple sequential steps that are logically connected with one another. The results found and shown in chapter 12 provide various insights on the quality of generated synthetic fraudulent transactions, the adversarial prowess of certain specific synthetic samples and the impact of these samples on the robustness of FDMs when added to a respective training dataset. This section will provide the reader with an overview of the most striking findings for each step of the process, along with their implications for our use case at bunq.

13.1. The quality of synthetic samples

The quality of the synthetic data generated by a CTGAN with a certain configuration can be interpreted in various ways, depending on the goals of the model designer. When the goal is to create synthetic data that is indistinguishable from the real data, data similarity metrics are a good indicator of quality. When the goal is to create synthetic data that is able to convey as much as information as the real data to a machine learning model, then machine learning utility metrics provide a good picture of the quality. Lastly, when the goal is to improve the degree of privacy preservation in the training dataset, then privacy preservation metrics are the go-to quality indicators. In this work, we consider all of the above to be a factor in the definition of synthetic data quality. The quality of the synthetic data is not determined as a stand-alone value, but rather its improvements or deterioration compared to the existing setting.

13.1.1. Similarity to the real data

Synthetic fraudulent samples that are near-indistinguishable from real fraudulent examples are desirable for a multitude of reasons. Not only does one expect synthetic data that resembles real data very closely to provide good input data for machine learning purposes, but also represent the complex way of human thinking found in terms of fraudulent behavior. On top of this, when human expert annotators are no longer able to distinguish between real and synthetic samples, we expect these samples to be a valuable addition to our existing knowledge base. After all, our supervised model knows no more than the knowledge held in our training dataset, which in turn is a result of human annotation. As explained in method chapter 11, the Logistic Regression Detection (LRD) metric gives us a good indication of the overall similarity of the synthetic data to the real data. Moreover, the Chi-squared (CS) and Inverted Kolmogorov-Smirnov D (IKS) statistic give us an idea of the similarity of the categorical and numerical columns respectively.

Sampling and reconstruction methods

To reiterate: rejection sampling will remove samples with impossible feature value combinations from the set of generated samples, based on a set of rules. Rejection sampling trades real-world realism and retaining membership to the joint distribution learned by the CTGAN for performance in terms of time. On top of this, as feature sets and their relationships get increasingly complex, one is at risk of having nearly all samples be rejected. Instead of outright rejecting samples, transformation sampling will instead make small changes to the feature values in order for them to follow the same set of predefined rules. Transformation sampling sacrifices a small degree of adherence to the joint distribution found by the model for an increase in time performance, as well as real-world realism. A few quick tests during the design phase of this project have shown that when generating samples using both methods using the same set of realism rules, transformation sampling generally outperforms rejection sampling in terms of run-time.

When considering figures 12.1, 12.2 and 12.3 one characteristic stands out immediately: the samples produced by rejection sampling show a significantly higher similarity to the real data than the ones produced by transformation sampling, according to all three metrics. Since both methods follow the same set of realism enforcing rules, one can only conclude that the difference similarity is the result of the transformation-sampling-based samples sacrificing adherence to the joint distribution in favour of not rejecting a sample outright. This difference is especially apparent in the IKS graphs, which can be explained by the fact that the majority of the features of our use-case are numerical.

Furthermore, both rejection and transformation sampling show a poor performance in terms of the LRD score, which implies that it is fairly easy for a logistic regression detection model to distinguish between the real and synthetic data. This is an interesting insight, as both the CS and IKS scores are fairly high, indicating that both the categorical and numerical columns, at least separately, represent the real data quite well. This discrepancy would imply that while the individual columns are a good representation of their real counterparts, the complete synthetic samples show a distinct difference from the real samples. In turn, this would imply that either the synthetic data partly fails to capture the relationships between features found in the real data or shows a specific distinction which is easily recognized by the LRD model. One potential cause for this is *mode collapse*, a common problem in the field of GANs. As explained in 4.3.2, mode collapse results in the generator only learning to generate a small set of output types. If this is the case, then this small number of distinctive outputs of the generator would be easy to recognize for the LRD model compared to a wide variety of output types.

Hyperparameter implications

Again considering figures 12.1, 12.2 and 12.3, we find some interesting relationships between different values for the hyperparameters that we consider to be influential to the quality of the produced synthetic samples. First of all, increasing the learning rate seems to have a detrimental effect on the similarity of the synthetic data. We can see a downwards trend for each of the likeness metrics as the learning rate increases. This relationship makes sense, since increasing the learning rate makes it harder for the model to effectively learn the more fine-grained properties of the training data. Interestingly, increasing the generator dimension also seems to have a detrimental effect on the similarity to the real data. For nearly all metrics, we see a network with a generator that has 128 residual layers outperform their 256 and 512 layered counterparts in terms of similarity. As we learned from section 4.3.2, increasing the generator dimension allows for the generator to learn more complex relationships, but comes at the cost of overfitting. Since one would expect a model overfitted to the training dataset to produce samples that have an over-exaggerated similarity to the real samples, these results are unexpected. Lastly, different values for the weight decay do not appear to impact the similarity in terms of any metric of the synthetic samples in a meaningful pattern.

While we consider the similarity to the real data to be an indicator of the quality of the synthetic samples, one should not be too hasty to consider this property to be the be-all and end-all. As an example: consider a synthetic dataset that is an exact copy of the real dataset. Trivially, this synthetic dataset has perfect similarity scores across the board, while providing none of the advantages of synthetic data such as providing meaningful samples of the minority class to improve class balance in a training dataset or to improve privacy preserved in the samples. Such an example would also be the result of extreme overfitting, which is far from desirable. For this reason, we will now consider the other two indicators of quality.

13.1.2. Machine learning utility

Since improving the robustness and overall performance of existing FDMs is one of the main goals of this work, analyzing and evaluating the machine learning utility of the synthetic samples is of utmost importance. Samples in the training dataset that lead to a poor performance of the machine learning model are not desirable in any scenario. For this reason, carefully crafted configurations to train CT-GANs have been explored to find which synthetic samples are effectively able to convey (parts of) the information held in fraudulent transactions.

Sampling and reconstruction methods
When comparing both sampling methods in figures 12.4, 12.5 and 12.6, the results show a clear difference in machine learning utility. Much like for the similarity metrics, rejection sampling seems to outperform transformation sampling in terms of machine learning utility as well, albeit by a lower margin. Comparing both methods to the baseline, the set of real fraudulent samples, we see a distinct difference in performance. After all, the real dataset is always expected to more effectively convey the information than their "counterfeit" counterparts. Still, FDMs trained on our synthetic samples, for both sampling methods, show fairly good results in terms of F1 score and ROC AUC. Rejection sampling, which shows to be the superior method in terms of machine learning utility, is able to produce FDMs with $F1 \approx 0.60$ and $ROC\ AUC \approx 0.80$. Compared to a model trained on the real samples, with $F1 \approx 0.80$ and $ROC\ AUC \approx 0.95$, this is of course a significant drop in performance. Still, considering the fact that the models trained on synthetic fraud samples did not have access to any real fraud samples, we consider such a performance to be quite good, especially considering the fact that the degree of privacy preserved in synthetic samples is significantly higher than in real samples. Lastly, as we expected, the likeness to the real data does seem to have an effect on the machine learning utility to some extent. While the margins of difference between machine learning utility of the sampling methods are smaller than the difference in likeness, both seem to follow a similar pattern.

Hyperparameter implications
One observation stands out immediately: none of the hyperparameters seem to have a significant impact on any of the FDMs trained on transformation-sampled datasets. It seems apparent that the changes made by transforming the samples are impactful in such a way that the smaller, more subtle changes from tuning the hyperparameters are effectively nullified. As one could expect, making slight alterations to the joint distribution learned by the CTGAN and then "breaking" this distribution when reconstructing the samples do not go hand in hand. For this reason, it makes sense that none of the hyperparameters significantly impact the transformation samples, unless the values of the hyperparameters become sufficiently small or large, for instance shown in figures 12.4a and 12.4b.

Considering rejection sampling, we see that the learning rate has a fairly small impact. Increasing the learning rate seems to slightly reduce the machine learning utility of the samples, albeit at a small, mostly insignificant rate. As opposed to the similarity, the generator dimension impacts the machine learning utility in an expected manner. Due to the fact that our fairly large feature set allows for a high complexity to be captured in the real samples, it turns out 128 residual layers does not suffice for the generator to learn most patterns and relationships found in the real data. On the other hand, when using 512 residual layers the model becomes too complex and overfits to the training dataset, which results in a lower machine learning utility of the produced synthetic samples. As it turns out, using 256 layers lies on the fine line between these two and achieves superior performance. Also contrasting the previous experiment, the generator decay does significantly impact the machine learning utility for rejection sampling. We can see that as the weight decay increases, the machine learning utility improves quite drastically. From this observation we can conclude that for low weight decay values the CTGAN tends to overfit to the training dataset, in turn producing training samples that hurt the generalizability of the FDM. Higher values for the weight decay, in turn, produce synthetic samples that more accurately convey fraudulent transactions in general.

13.1.3. Privacy preservation
As discussed in great detail in this work, privacy can be interpreted in a wide variety of ways. Considering our first metric, the distance to the closest record, we remember that it gives us an indication of the degree in which the synthetic dataset leaks real information found in the training dataset. When we find a DCR of 0 it means that the synthetic data directly leaks data from the real dataset. As Park et al. state in their work, we strive for a high average distance along with a low standard deviation [76]. Such a combination would indicate that the overall distance between the records of both datasets is high, meaning that in general very little real information is leaked. Combined with a low standard deviation this means that no records exist that leak real information. Tables 12.1 to 12.6 show that for nearly all configurations and sampling methods the average distance is relatively high compared to the DCR and standard deviation. This would indicate that the aforementioned relationship holds for the base samples generated by the CTGAN.

Sampling and reconstruction methods
The main distinction that we find in tables 12.1 to 12.6 is that while transformation sampling shows to have significantly higher values for the DCR, it gets outperformed by rejection sampling in terms of the average and standard deviation. What this means is that samples created using rejection sampling on average preserve privacy significantly better than transformation sampling, while transformation sampling has a better lower bound on the sample that is the closest to the real dataset. From this we can conclude that both rejection sampling and transformation sampling provide unique advantages in terms of privacy preservation:

- **Rejection sampling** provides a generally good degree of privacy by providing samples that are significantly different from the real data, while having a low standard deviation which implies that there exist no or very few samples that directly leak real data.
- **Transformation sampling** provides the same advantages as rejection sampling, albeit to a lower degree. Furthermore, since transformation sampling shows a significantly higher DCR on average, it provides a better lower bound in terms of how close the sample closest to the real dataset is. In other words, the sample with the highest chance of leaking real data is significantly less useful to an adversary when using transformation sampling compared to rejection sampling.

Hyperparameter implications
The most striking relationship that we find between different values for the hyperparameters and privacy preservation is that none of the parameters have a significant effect on the samples generated by transformation sampling. This can, again, be explained by the fact that transformation sampling changes the samples in a more significant manner than the subtle changes brought about by changes in the hyperparameters can. More interestingly, we find that changes in hyperparameters **do** effect the samples generated by rejection sampling. Increasing the learning rate leads to a higher DCR and average distance, while the standard deviation goes up. This relationship is to be expected, as a higher learning rate favours exploration and in turn the variety in the samples generated. A lower learning rate produces samples that are generally more similar to the training data and thus lower values and deviation in the produced samples are expected. Similar to the learning rate, the generator dimension parameter mostly affects the samples produced by rejection sampling. Increasing the generator dimension, in other words allowing the generator to learn more complex patterns at the risk of overfitting, leads to a slightly higher DCR, which means less real data is directly leaked through the closest record. In contrast, the standard deviation increases slightly while the average distance decreases significantly. We can explain these results by concluding that for the highest value of the generator dimension, the generator overfits to the training data to the degree that the average distance decreases significantly. Lastly, the generator decay does not seem to affect the degree of privacy in terms of distance for both rejection sampling and transformation sampling. Only when the generator decay gets sufficiently high, in this case $1.5e-6$ we find a significant improvement in terms of the average distance of the samples generated by both methods. This leads us to assume that even at the moderate, default values for the generator there exists overfitting to some degree. As higher values for generator decay aim to prevent the model from overfitting, this could explain the significant increase in average distance for these values. It is likely that this overfitting is the result of a fairly small training dataset. These findings are corroborated by the findings of the previous experiment, where we can see the machine learning utility provided by the samples increase significantly as the generator decay increases.

13.1.4. Adversarial attacks

While distance metrics provide us with easy to understand insights of the privacy preservation of synthetic samples, they fall short in helping us understand how smarter adversaries are able to use the synthetic data to achieve their goals. While distance measures help us understand to what degree real data is leaked through the synthetic samples, we can use the data collected from performing adversarial attacks on the synthetic data to learn to understand to what degree adversaries are able to extract sensitive information from the synthetic samples. To this end, we have explored how likely an adversary is to find the true values of two sensitive attributes: user gender and user nationality.

Considering figures 12.7 and 12.8, we find one striking difference between the two attributes: the CAP attacker on average has almost double the accuracy of detecting the users nationality than it has of the users gender. One explanation for this difference is the fact that the CAP attacker has more meaningful features to use to determine the value for user nationality. One could argue that user language, current country of residence and and the transaction being an international payment contain valuable information for determining user nationality. Comparing these features to the features used to determine user gender, the most relevant being the age of the user and the country of origin of the user (in some niche cases), it is easy to see why the model has an easier time determining the nationality of the user. On top of this, the results show that generally the CAP attacker has a higher accuracy when attacking datasets generated by rejection sampling. This makes sense, as discussed above, the rejection samples are on average more similar to the real samples. When considering the effect of the hyperparameter configurations on the accuracy of the CAP attacker, we find that the learning rate and generator dimension affect the user nationality to a more significant degree and pattern compared to user gender. We can explain this by the fact that the features used to determine the user gender used by the CAP attacker hold less relevant data. The implications of these shortcomings are significantly more impactful than the subtle changes introduced by different hyperparameter configurations, which in turn leads to a less significant relationship between the two.

From figure 12.7 we can see that as the learning rate increases, the CAP attackers accuracy decreases. This can be explained by the fact that increasing the learning rate promotes variety in the samples produced, rather than the similarity of the average sample to the training data. As a result, the CAP attacker has a more difficult time learning the patterns present in the real dataset from the synthetic data when the learning rate increases, which in turn reduces its accuracy. Furthermore, we can see a significant relationship between the generator dimension and the accuracy of the CAP attacker. We can see that as the generator dimension increases, the accuracy of the CAP attacker increases as well. Similar to the distance metrics, a significant degree of privacy preservation is lost as the result of using an inordinately high value for the generator dimension. Again, we blame overfitting as a result of these high values, as overfitting to the training data results in synthetic samples that are disproportionally similar to the real samples.

13.1.5. Privacy preservation vs. machine learning utility

In the last round of experiments on the quality of the generated samples, we consider the trade-off introduced by privacy preservation and machine learning utility. Figures 12.9a and 12.9b show the relationship between these two quality indicators. The results that we find are as expected: as the degree of privacy in terms of DCR increases, we can see that the machine learning utility of the samples decreases. Even though we find that this relationship exists, we should not draw conclusions too hastily. While both the F1 score decrease as a result of an increase in privacy preservation, the absolute differences in terms of the scores is not very high. When considering table 12.7, we can see quite a difference between the average DCR found in the set of privacy preserving samples compared to the random samples. When comparing both edge cases, we find a percentage loss in F1 score of $(0.828 - 0.803)/0.828 * 100 = 3,01\%$ and in ROC AUC score of $(0.957 - 0.941)/0.957 * 100 = 1.67\%$. When compared to the percentage increase in real data leakage of $(0.122-0.086)/0.122*100 = 29.5\%$, we find that this privacy trade-off is actually quite advantageous. Still, we remark that the drop in machine learning utility is a direct result of the loss in variety of the selected samples. While the top privacy preserving samples might provide a decent privacy guarantee, the samples lack in variety of information that is conveyed to the FDM and in turn hurts its generalizability.

Overall, we conclude that while the synthetic samples provide unique advantages in terms of privacy, a higher degree of privacy than currently achieved is desirable before being able to be made public in any way. Still, our results provide interesting insights with respect to the degree of control that a model designer has with respect to the privacy implications of the generated synthetic samples.

13.2. Adversarial prowess of synthetic samples

While the previous experiments have focused on the relationship between sampling methods, CTGAN hyperparameters and the quality of the produced synthetic samples, the adversarial prowess experiments shift the focus towards how one filters the samples which are expected to be UUs to the model and in turn hold the most valuable information for increasing the robustness of our FDM to false negatives. To achieve this, we aim to figure out a quantification method which describes how well synthetic samples are expected to beat $FDM_{default}$. After the previous rounds of experiments, we assume our synthetic samples qualify as bona fide fraudulent transactions. Operating under this assumption, we expect a robust FDM to be able to properly classify these synthetic transactions as fraudulent. However, using the method explored in experiment 13.2.2, we find that filtering the synthetic samples is a non-trivial task and that using this method we are able to find the samples with the highest adversarial prowess. In other words, we are able to distinguish between samples that are highly likely to be a false negative to $FDM_{default}$ and those that are not.

13.2.1. General findings

We consider the adversarial prowess as expressed by the percentage of false negatives found by $FDM_{default}$ out of all the (synthetic) fraudulent samples in any dataset. As a baseline, we compare the results to real fraudulent transactions. Because the synthetic samples will never be a perfect representation of real fraudulent samples, we always expect more false negatives to be found for synthetic datasets than for real datasets. For this reason, to accurately grasp the advantages introduced by our adversarial filtering method, we also include a baseline of random synthetic samples. If the adversarial filtering method is not able to produce samples that are significantly more likely to be a false negative to $FDM_{default}$, then the filtering method would pose no advantages over using just any synthetic sample.

As can be seen from figures 12.13a to 12.13c, the adversarial filtering method does produce samples that are significantly more difficult for the model to classify as fraud. In other words, under the assumption that the synthetic fraudulent transactions qualify as fraudulent samples, this method finds not-yet-seen fraudulent transactions that would fly under the radar of $FDM_{default}$, in other words false negatives. As expected, random synthetic samples also produce a significantly higher number of false negatives compared to the baseline of real fraud, for the reason mentioned above. For most configurations of weight parameters, which we will discuss soon, random synthetic samples are on average $\sim 25\%$ more likely to a false negative compared to real fraudulent samples. Adversarially filtered synthetic samples are shown to be $\sim 50\%$ more likely to be falsely classified as legit than a real fraudulent sample would.

13.2.2. Implications of weight parameters

As introduced in method section 9, the filtering method makes use of both data-driven and human-expert-driven methods to calculate a score representing the adversarial prowess of each synthetic sample. In experiment we compare the implications of different values of each of these parameters. We find the relationships between the parameters used in in the adversarial filtering system in figures 12.10a to 12.12a. First off, it turns out that both weights for numeric uniqueness and boolean uniqueness values have very little influence on the adversarial success of the synthetic samples. To a degree, this makes sense. We use these uniqueness values as a base for our success score, but neither of these base values are directly related to adversarial success. Any transaction, fraudulent or legit, can possess unique feature values such as a country not often seen in other transactions or a very high monetary amount, for instance when someone buys a house in a different country. Secondly, from figure 12.12a we can see that a higher value of the Shapley weight has a positive influence in terms of the adversarial strength of the filtered samples. As expected, selecting samples based on how ”bad” the $FDM_{default}$ is expected to be at classifying them, the better they perform as adversarial examples. Thirdly, and more unexpectedly, the expert perceptibility score has a negative relationship with the adversarial strength of the filtered samples. We expected that since human experts manually label each transaction which, in turn, influences the learning process of the FDM, selecting samples based on their likeliness to be noticed by a human expert would positively affect the adversarial success of a sample. As it turns out, this parameter has little effect on the adversarial success and even reduces it at higher values. Lastly, we can see an inverse relationship between the categorical cardinality bound value and the adversarial success score. To reiterate: the higher this value is, the higher the impact of unique categorical values is on the success score calculation. Evidently, limiting the impact of such unique values for categorical features turns out to positively impact the adversarial success of the filtered samples. After all, an unique value for a categorical value might be an interesting indicator of a fraudulent transaction, but should not have a disproportional impact on any classification.

13.3. Training models on augmented datasets

Lastly, we consider the final and most important round of experiments. Whereas the previous rounds of experiments have served to allow us to understand how to produce high-quality, privacy-preserving and adversarially strong synthetic samples, this experiment aims to determine whether adding such samples to the training dataset actually improves the robustness and overall performance of the FDM. When considering figures 12.14a to 12.19b we can see that the different ways each training dataset is set up significantly impacts the performance of the respective FDM. After all, the training dataset holds all of the information that the FDM is able to learn. The most striking and main accomplishment of this work can be found when comparing figures 12.14b and 12.17b. These figures represent the existing scenario baseline ($FDM_{default}$) and the scenario where we augment the training dataset with adversarially filtered synthetic fraud samples in an equal amount to the number of real fraudulent samples. We find that model F is able to correctly classify 7% more never-seen-before fraudulent transactions than its un-augmented counterpart, model A. From figure 12.18b we find that even random synthetic fraudulent samples help the model to generalize better. Comparing figure 12.18b to 12.17b, we learn that the adversarial filtering method does in fact produce synthetic fraudulent samples that help the model generalize to a higher degree.

As expected, this improvement does come at the cost of an increased number of false positives. A probable cause of this increase, is the fact that adversarial examples are selected based on the mindset of theorem 2. Since false positives are grouped with all other non-false-negative results of the FDM when constructing the labels of the FVAM and such samples are penalized by our adversarial filtering system, it is plain to see that the selected adversarial samples will be of little help to provide useful information on false positives. At first glance, when considering figures 12.14a and 12.17a, an increase of 1% does not seem significant. Here we must remember that the large majority of the transactions we encounter in the wild belong to this majority class: the legit transactions. In the last section of the discussion chapter, we consider this trade-off to find out whether an implementation of this system would provide any benefits in the complete cost picture of financial fraud.

From figures 12.15a, 12.15b, 12.16b and 12.16b we find that only the synthetic fraudulent samples, adversarially filtered or not, hold significantly less information than the real fraudulent samples. This of course makes sense, as the real samples are considered to be as good as it gets. These findings, combined with the findings described above, teach us that the (adversarially filtered) synthetic samples provide a strong addition to the existing fraudulent samples, rather than an effective stand-alone representation of the knowledge found in fraudulent samples. Another interesting angle where such a benefit shows can be found when considering figures 12.19a and 12.19b, we see a rather unexpected jump in performance in terms of the number of false positives compared to the baseline model A. On top of this, it comes at the relatively small price of a small increase in false negatives. To reiterate: model I is trained on the real fraudulent samples, double the number of random synthetic samples and all of the real legitimate samples. As it turns out, providing the model with a notable number of extra synthetic samples from the minority class helps the model generalize as well.

13.3.1. Classifier thresholds

When the training process of a model is done, the H2O framework automatically calculates an optimal threshold for the classifier based on the chosen performance metric. The metric that we use is the F1 score and as found in the previous discussion section, the resulting models perform quite well. However, the optimal threshold might not always fit the use case of the model designer. This means that exploring different thresholds to, in turn, control the number of false positives or false negatives is an interesting venue.

In figure 12.20 we find the performance of model F in terms of the number of classifier mistakes, both false positives and false negatives, compared to our baseline models. Following the red and salmon colored lines, representing the number of false positives and false negatives respectively, we find that we are able to bring the number of false positives down to the level of the default situation ($FDM_{default}$), represented by the blue model A line, while still retaining a lower number of false negatives. This area of interest starts at a threshold of roughly 0.22, where the red, blue and salmon lines meet and ends at a threshold of roughly 0.26, where the salmon and light blue lines cross. Based on the current needs of the transaction monitoring system, ranging from minimizing false positives to decrease compliance workload to minimizing false negatives to reduce costs incurred by fraud committed, the model designer is able to select any threshold in the range $[0.22; 0.26]$ without incurring any extra costs in either false positives or false negatives when compared to model A.

To consider the overall performance of each model, represented by the F1 score, we turn to figure 12.21. As it turns out, model F outperforms model A in terms of the F1 score in an even larger threshold range than $[0.22; 0.26]$. This implies that the model designer is able to make even more radical design choices in terms of choosing the model threshold without dropping below the performance of model A. This is a very interesting insight, as the environment of fraud prevention is ever changing. At any point might the costs incurred by either false positives or false negatives shift drastically, requiring an immediate answer. As this figure shows, adapting model F allows for such a drastic change without dropping below the optimal performance threshold of the default model. In a normal situation however, one would opt for the optimal F1 score, represented by the highest point of model F shown in figure 12.21 at a threshold value of roughly 0.2.

13.3.2. Reducing false positives

Since the false negative generalization benefits of our most promising model, model F, come at the price of an increased number of false positives, effort was made to try reduce this number through external means. This effort is two-fold: using weights in our training dataset to more accurately represent samples that have shown to be difficult to label as being either fraudulent or legit and implementing a system that filters out repeated false positive hits of the FDM.

Weights on edge-cases

From figures 12.22b and 12.23b we find that while the addition of weights to these cases helps to reduce false positives, it does considerable harm in terms of false negatives. In the previous experiment we found that, when considering our top model F, we are able to trade 1% more false positives for 6.4% less false negatives. The numbers found when introducing weights are 0.6% less false positives in exchange for 2.1% more false negatives. Depending on the needs of the transaction monitoring system, this trade-off could be considered worth the effort. In a normal situation however, the drop in model performance as expressed by both the F1 and F2 scores of the weighted model compared to its unweighted counterpart make the addition of weights undesirable.

13.4. Real world implications of the found results

Lastly, to translate the obtained results into a more easily understandable format, we ask the question: how much monetary loss could be mitigated by implementing this method in a real-world setting? We find that, even though the proposed method leads to a significant increase in the raw number of false positives compared to the decrease in the raw number of false negatives, due to the relatively enormous impact of false negatives compared to false positives, there is still a significant improvement in terms of mitigated financial loss. Our calculations find that just a bit more than $1/6$th of these losses are mitigated when employing the proposed method. In existing literature however, as we have seen in section 5.1, the cost ratios between false negatives and false positives are less extreme than in our concrete example, ranging from $143 : 1$ to $82 : 1$, whereas our ratio was found to be $238 : 1$. Based on the ratios found in the literature, the estimated reduction in costs ranges from $1/6$ to $1/11$ to $1/16$, which are all a sizeable reduction in raw monetary losses.

14 Conclusion

In this thesis we have thoroughly explored the field of financial fraud along with the challenges imposed on automated decision making. A particularly close look was taken at the relationship between these challenges and the process of training an automated decision maker in the form of a supervised learning model. Based on our findings we have created a machine learning pipeline that uses a wide range of available knowledge in the field to address these challenges. As the first step of this pipeline we have created two methods to generate high-quality synthetic fraudulent samples and explored many ways in which the CTGAN framework can be used further increase this quality. Secondly, we have introduced a novel filtering method to find unknown unknowns in the synthetic samples based on data-driven and human-expert-driven methods. Using this method to select the top samples from these generation methods, we have found that augmenting training datasets with such samples provides considerable advantages in terms of the robustness of the trained model to never-seen-before fraudulent behavior. Because these advantages come at the cost of small degree of losses in terms of false positives, we have created and tested two standalone methods to mitigate said losses. Finally, we show that applying this method in a real-world scenario could result in a mitigation of 1/6 of the monetary losses incurred by financial fraud, and based on literature in the field, perhaps even higher.

Future improvements on this work can take shape in multiple different sections of the pipeline. Firstly, a more in-depth look could be taken at comparing the synthetic samples generated by different frameworks. During the design phase of this project CTGAN was selected for its promising aspects, easy of implementation in Python and promising initial results. Still, especially since the field of GANs and synthetic data generation is an active field of research, exploring different methods is an interesting venue. Secondly, more research in the direction of the privacy guaranteed by the synthetic samples is needed to create samples that are sufficiently safe to release to the public. While the synthetic samples generated in this work pose interesting and unique advantages over using real samples, the degree of privacy achieved currently is insufficient to be released to the public. Different venues, for instance differential privacy, could be explored to further improve the degree of privacy.

Thirdly, during the evaluation of the performance of the FDMs on different datasets, a static set of training datasets was explored. It would be an interesting direction of research to find if an optimal ratio between random synthetic, adversarial synthetic and real samples exists. Our results show that both random synthetic as well as adversarial synthetic samples provide unique examples to the generalizability of the trained FDM, which implies that some global optimum exists with regards to the ratios between the three sample types. Lastly, as mentioned many times in this work, the field of fraud detection is colourful in nature. For this reason, using a multi-label classification approach instead of a binary classification approach could bring interesting insights to the table. The differences between different types of fraud are significant in some cases, which implies that using a binary classification approach is an oversimplification. It is possible that using multiple labels for different types of fraud is able to capture unique relationships compared to a binary approach. For this reason, exploring a multi-label approach in tandem with the contributions made by this work could be an interesting venture.

Bibliography

[1] Josh M Attenberg, Pagagiotis G Ipeirotis, and Foster Provost. Beat the machine: Challenging workers to find the unknown unknowns. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[2] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*, 2019.

[3] Gagan Bansal and Daniel S Weld. A coverage-based utility model for identifying unknown unknowns. *AAAI, 2018*, 2018.

[4] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[5] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4502–4511, 2019.

[6] Messod D Beneish and Patrick Vorst. The cost of fraud prediction errors. *Kelley School of Business Research Paper*, (2020-55), 2020.

[7] Betaalvereniging NVB. Persbericht: Sterke stijging cybercriminaliteit leidt tot meer schade, 2021.

[8] Betaalvereniging Nederland. Factsheet betalingsverkeer 2020, 2020.

[9] Concha Bielza and Pedro Larranaga. Discrete bayesian network classifiers: A survey. *ACM Computing Surveys (CSUR)*, 47(1):1–43, 2014.

[10] Hendrik Blockeel and Jan Struyf. Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research*, 3(Dec):621–650, 2002.

[11] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[12] Louis Brandeis and Samuel Warren. The right to privacy. *Harvard law review*, 4(5):193–220, 1890.

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[15] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The guardian*, 17:22, 2018.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[17] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[18] Michael J Cerullo and Virginia Cerullo. Using neural networks to predict financial reporting fraud: Part 1. *Computer Fraud & Security*, 1999(5):14–17, 1999.

[19] 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*, pages 785–794, 2016.

[20] Yeounoh Chung, Peter J Haas, Eli Upfal, and Tim Kraska. Unknown examples & machine learning model generalization. *arXiv preprint arXiv:1808.08294*, 2018.

[21] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *arXiv preprint arXiv:1909.03683*, 2019.

[22] CM Cuadras and C Arenas. A distance based regression model for prediction with mixed data. *Communications in Statistics-Theory and Methods*, 19(6):2261–2279, 1990.

[23] De Nederlandsche Bank. Guideline on the anti-money laundering and anti-terrorist financing act and the sanctions act, 2011.

[24] Bernhard Debatin, Jennette P Lovejoy, Ann-Kathrin Horn, and Brittany N Hughes. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of computer-mediated communication*, 15(1):83–108, 2009.

[25] Patricia M Dechow, Weili Ge, Chad R Larson, and Richard G Sloan. Predicting material accounting misstatements. *Contemporary accounting research*, 28(1):17–82, 2011.

[26] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*, pages 715–733. Springer, 2020.

[27] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[28] Thomas G Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

[29] Alan Doig. *Fraud*. Crime and society series. Taylor and Francis, 2013. ISBN 9781843926115,1843926113,978-1-84392-173-8,1-84392-173-1,978-1-84392-172-1,1-84392-172-3. URL `http://gen.lib.rus.ec/book/index.php?md5=680c79554bfa27532007de9af1a646f0`.

[30] Jose R Dorronsoro, Francisco Ginel, C Sgnchez, and Carlos S Cruz. Neural fraud detection in credit card operations. *IEEE transactions on neural networks*, 8(4):827–834, 1997.

[31] Jacqueline M Drew and Cassandra Cross. Fraud and its prey: Conceptualising social engineering tactics and its impact on financial literacy outcomes. In *Financial Literacy and the Limits of Financial Decision-Making*, pages 325–340. Springer, 2016.

[32] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

[33] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[34] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[35] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

[36] Edward A Feigenbaum. Expert systems in the 1980s. *State of the art report on machine intelligence. Maidenhead: Pergamon-Infotech*, 1981.

[37] Eliezer M Fich and Anil Shivdasani. Financial fraud, director reputation, and shareholder wealth. *Journal of financial Economics*, 86(2):306–336, 2007.

[38] Janick V Frasch, Aleksander Lodwich, Faisal Shafait, and Thomas M Breuel. A bayes-true data generator for evaluation of supervised and unsupervised learning methods. *Pattern Recognition Letters*, 32(11):1523–1531, 2011.

[39] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[40] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[41] Christopher Frye, Damien de Mijolla, Tom Begley, Laurence Cowton, Megan Stanley, and Ilya Feige. Shapley explainability on the data manifold. *arXiv preprint arXiv:2006.01272*, 2020.

[42] Xiuju Fu, Lipo Wang, Kok Seng Chua, and Feng Chu. Training rbf neural networks on unbalanced data. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, volume 2, pages 1016–1020. IEEE, 2002.

[43] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.

[44] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[45] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[46] Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, 2019.

[47] Sara Hajian, Josep Domingo-Ferrer, and Antoni Martinez-Balleste. Discrimination prevention in data mining for intrusion and crime detection. In *2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pages 47–54. IEEE, 2011.

[48] Thomas Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122, 1996.

[49] Andrew J. Hawkins. Uber settles claims that it mishandled private information about users and drivers. August 2017.

[50] Markus Hittmeir, Andreas Ekelhart, and Rudolf Mayer. On the utility of synthetic data: an empirical evaluation on machine learning tasks. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.

[51] Markus Hittmeir, Rudolf Mayer, and Andreas Ekelhart. A baseline for attribute disclosure risk in synthetic data. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 133–143, 2020.

[52] Ned Horning et al. Random forests: An algorithm for image classification and generation of continuous fields data sets. In *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan*, volume 911, 2010.

[53] Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

[54] Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*, 2019.

[55] Michael Kearns. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*, 1988.

[56] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature biotechnology*, 26(9): 1011–1013, 2008.

[57] Efstathios Kirkos, Charalambos Spathis, and Yannis Manolopoulos. Data mining techniques for the detection of fraudulent financial statements. *Expert systems with applications*, 32(4):995–1003, 2007.

[58] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, 2(1):1–15, 2000.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[60] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. *arXiv preprint arXiv:1610.09064*, 2016.

[61] Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. Adversarial filters of dataset biases. 2019.

[62] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[63] Hongya Lu, Haifeng Wang, and Sang Won Yoon. A dynamic gradient boosting machine using genetic optimizer for practical breast cancer prognosis. *Expert Systems with Applications*, 116: 340–350, 2019.

[64] Joseph Luft and H Ingham. The johari window: a graphic model of awareness in interpersonal relations. *Human relations training news*, 5(9):6–7, 1961.

[65] John F Magee. *Decision trees for decision making*. Harvard Business Review, 1964.

[66] D Suja Mary and M Suriakala. Detection of causative attack and prevention using cap algorithm on training datasets. In *International Conference on Inventive Computation Technologies*, pages 431–440. Springer, 2019.

[67] John Mingers. Expert systems—rule induction with statistical data. *Journal of the operational research society*, 38(1):39–47, 1987.

[68] Openbaar Ministerie. Fraudemonitor 2017 en 2018, 2018. URL `https://www.rijksoverheid.nl/documenten/rapporten/2019/07/19/tk-bijlage-fraudemonitor-2017-en-2018`.

[69] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[70] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[71] Giang H Nguyen, Abdesselam Bouzerdoum, and Son L Phung. A supervised learning approach for imbalanced data sets. In *2008 19th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[72] Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable ai: Hybrid human-machine analyses for characterizing system failure. *arXiv preprint arXiv:1809.07424*, 2018.

[73] United States Department of Defense (defense.gov). Defense.gov news transcript: Dod news briefing - secretary rumsfeld and gen. myers, 2002. URL `https://archive.defense.gov/Transcripts/Transcript.aspx?TranscriptID=2636`.

[74] De Nederlandse Overheid. Wetboek van strafrecht, july 2021. URL `https://wetten.overheid.nl/BWBR0001854/2021-07-01`.

[75] WA Parent. Privacy, morality, and the law. In *Privacy*, pages 105–124. Routledge, 2017.

[76] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*, 2018.

[77] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE, 2016.

[78] Simon J Pittman and Kerry A Brown. Multi-scale approach for predicting fish species distributions across coral reef seascapes. *PloS one*, 6(5):e20583, 2011.

[79] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[80] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3): 221–234, 1987.

[81] J Ross Quinlan. Decision trees and decision-making. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):339–346, 1990.

[82] J Ross Quinlan and Ronald L Rivest. Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248, 1989.

[83] Arjan Reurink. Financial fraud: a literature review. *Journal of Economic Surveys*, 32(5):1292–1325, 2018.

[84] Lior Rokach and Oded Maimon. Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer, 2005.

[85] Cicero Nogueira dos Santos, Youssef Mroueh, Inkit Padhi, and Pierre Dognin. Learning implicit generative models by matching perceptual features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4461–4470, 2019.

[86] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[87] Robert E Schapire. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pages 149–171, 2003.

[88] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *ICML*, volume 2, page 6. Citeseer, 2000.

[89] Pranab K Sen et al. Gini diversity index, hamming distance, and curse of dimensionality. *Metron-International Journal of Statistics*, 63(3):329–349, 2005.

[90] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[91] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.

[92] Ryan Singel. Netflix spilled your brokeback mountain secret, lawsuit claims. *Threat Level (blog), Wired*, 2009.

[93] James Charles Smith. The structural causes of mortgage fraud. *Syracuse L. Rev.*, 60:473, 2009.

[94] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[95] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[96] Herman T Tavani. Informational privacy, data mining, and the internet. *Ethics and Information Technology*, 1(2):137–145, 1999.

[97] Samir Touzani, Jessica Granderson, and Samuel Fernandes. Gradient boosting machine for modeling the energy consumption of commercial buildings. *Energy and Buildings*, 158:1533–1543, 2018.

[98] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[99] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

[100] Jarrod West and Maumita Bhattacharya. Intelligent financial fraud detection: a comprehensive review. *Computers & security*, 57:47–66, 2016.

[101] Eyal Winter. The shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.

[102] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503*, 2019.

[103] Lei Xu et al. *Synthesizing tabular data using conditional GAN*. PhD thesis, Massachusetts Institute of Technology, 2020.

[104] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480, 2009.

[105] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018.

[106] Bin Zhou, Yi Han, Jian Pei, Bin Jiang, Yufei Tao, and Yan Jia. Continuous privacy preserving publishing of data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 648–659, 2009.