Robustness and Observational Fairness in Dynamic Environments

Tiago Franco de Melo Pinto

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s):	Dr. Pedro dos Santos Saleiro da Cruz
	Prof. Mário Alexandre Teles de Figueiredo
	Sérgio Gabriel Pontes de Jesus

Examination Committee

Chairperson:

Supervisor:

November 2023

”Para ser grande, sê inteiro: nada
Teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.”
- *Para ser grande, sê inteiro*, Ricardo Reis

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

To my original supervisors, Pedro Saleiro and Mário Figueiredo for guiding me through the process of conducting this research, and to the one supervisor who was with me every day, listening to my ideas and enjoying my small victories, Sérgio Jesus.

To the remainder of Feedzai's research department, who taught me a lot and allowed me to become a better researcher. A special thanks to my fellow interns, who shared all the ups and downs of this journey with me.

To my mother Ana Paula, my father Paulo, and my sister Inês, whose love supported me all the way to this accomplishment. Without them, I couldn't have made it.

To the remainder of my family, who always believed in me. A special thanks to my late grandfather António, who would've been proud of me becoming an Electrical Engineer just like him.

To my friends, from whichever background, for allowing me to be myself even when work seemed overwhelming.

To all the researchers conducting work towards fair and robust AI techniques, as they are the ones contributing to a safer and more equitable future.

Resumo

A aprendizagem automática começa a ter um maior leque de aplicações, nomeadamente cenários de decisões de alto risco como finanças e saúde, onde erros podem ter grande impacto na vida de indivíduos. A responsabilidade acrescida nestes cenários requer testes minuciosos para garantir um funcionamento correto dos modelos após o seu lançamento. Consequentemente, testes de robustez surgem como um aspeto central na fase de desenvolvimento. Este estudo foca-se em deteção de fraude financeira, uma área dinâmica repleta de perturbações que obstruem o comportamento previsto dos modelos, tornando a análise de robustez mais difícil mas indispensável. Devido às implicaçoes de alto risco desta indústria, as decisões têm inflûencia profunda na vida de indivíduos, realçando a necessidade de garantir que algoritmos não discriminam ninguém com base em fatores como género e raça. Então, a nossa abordagem conecta os tópicos de robustez, ambientes dinâmicos e justiça algorítmica, introduzindo uma perspetiva inovadora de robustez que tem em conta a perturbação específica em escrutínio e que considera tanto desempenho como justiça algorítmica. Guiados por esta visão, apresentamos uma taxonomia de perturbações que engloba desde problemas relacionados com dados até ataques adversariais, e sugerimos um método estruturado para avaliar a robustez de modelos contra estas perturbações. Subsequentemente, aplicamos estas perturbações a três conjuntos de dados tabulares e expomos a falta de robustez de modelos populares. Também desenvolvemos um modelo treinado para ser robusto e usamos o nosso método para demonstrar que tem maior robustez do que o seu predecessor.

Palavras-chave: ambientes dinâmicos, aprendizagem automática, ataques adversariais, dados tabulares, justiça algoritmica, robustez

Abstract

Machine Learning (ML) is starting to find its way into a broader range of applications, including high-stakes decision-making settings such as finance and healthcare, where errors can heavily impact individuals' lives. The added responsibility in these scenarios requires thorough testing to guarantee correct model behavior after deployment. Consequently, robustness testing emerges as a core aspect of the development process. This study focuses on financial fraud detection, a dynamic landscape replete with a myriad of perturbations that obstruct the model's intended behavior, making robustness analysis a difficult yet indispensable endeavor. Given the high-stakes implications of the ML application, decisions profoundly influence individuals' lives, emphasizing the necessity of ensuring that algorithms do not discriminate based on factors such as gender, race, and the like. Hence, our approach bridges the realms of robustness, dynamic environments, and fairness, introducing a novel perspective on robustness that accounts for the specific perturbation under scrutiny, considering both performance and fairness. Guided by this vision, we present a taxonomy of perturbations, ranging from data-related issues to adversarial attacks, and suggest a framework tailored to evaluate the robustness of models against these perturbations. Subsequently, we put this framework into action by conducting tests on three distinct tabular datasets, applying those perturbations, and exposing popular models' lack of robustness. We also build a model with robustness concerns and employ our framework to demonstrate that it is more robust than its predecessor.

Keywords: adversarial attacks, algorithmic fairness, dynamic environments, machine learning, robustness, tabular data

Contents

List of Tables

List of Figures

Nomenclature

ACS	American Community Survey
AI	Artificial Intelligence
AUC	Area Under the Curve
BA	Boundary Attack
BAF	Bank Account Fraud
CI	Confidence Interval
FNR	False Negative Rate
FPR	False Positive Rate
GAN	Generative Adversarial Network
HSJA	HopSkipJump Attack
i.i.d.	Identically and Independently Distributed
LGBM	LightGBM (Gradient Boosted Machine)
ML	Machine Learning
NaN	Not a Number
NN	Neural Network
PSR	Perturbation Size Ratio
ROC	Receiver Operating Characteristic
SRR	Success Rate Ratio
PPV	Positive Predictive Value (Precision)
TPR	True Positive Rate

Chapter 1

Introduction

Artificial intelligence (AI) and Machine Learning (ML) are becoming a central tool in society. Their usefulness is growing with the amount of data available and the increasing computational power of our devices. The range of applications for this technology is no longer limited to simple tasks such as movie recommendations, but it is now also used in high-stakes decision scenarios. Said scenarios can be, for example, in financial services [1], in healthcare [2], and in criminal justice [3].

ML algorithms frequently operate within dynamic and ever-shifting environments, replete with a myriad of perturbations that pose a considerable challenge to their ability to maintain their intended behavior [4]. In this context, the need for thorough testing becomes evident, ensuring that models do not experience substantial performance degradation in the face of these perturbations [5]. Particularly within high-stakes settings, where errors bear a deep influence on individuals' lives, an additional layer of scrutiny is indispensable. Consequently, robustness testing emerges as a core aspect of the development process [6, 7]. This evaluation effort becomes essential to avoid undesirable scenarios that may arise due to the model's response in these stressful situations.

One of the applications where machine learning is thriving is financial fraud detection. Examples of financial fraud are money laundering, scams, and impersonation. Correctly distinguishing a legitimate action from a fraudulent one as fast and accurately as possible is a high-value asset in the financial industry, resulting in large monetary investments in solutions for this task [8]. Fraud detection is a high-stakes ML application where fairness is of the utmost importance, as people's lives are directly affected by the predictions, and should not be discriminated against based on sensitive protected attributes such as race or gender. Examples of consequences are denying access to bank accounts, blocking credit cards, or incurring monetary losses. Moreover, fraud detection is highly dynamic and adversarial, featuring complex interactions between the models and the environments, as well as shifts in the statistics of the data. Fraudsters are adaptive and always evolving their methods, attempting to be more successful, by increasing their effectiveness and profit. In this adversarial setting, fraudsters produce inputs that try to mimic regularly generated data but are wrongly classified by the targetted model. Additionally, data distributions can change naturally over time (e.g., due to a pandemic, such as COVID-19, more transactions were done online than with a physical card), or due to various perturbations, and the available fraud labels used by the ML algorithms can be noisy [9].

Therefore, ML models to detect fraud must not only be performant and fair but also robust to adversarial attacks and time-changing conditions. The most common way to deal with a dynamic environment is to constantly retrain the model with more recent data, thus reducing the systems' degradation in performance due to the changing environments. Unfortunately, this approach is resource-intensive, as it usually requires re-computing the entire optimization path. Furthermore, it is even unfeasible in scenarios such as the fraud detection industry, where the most recent data be cannot used due to the delay in the labeling process. By building robust models, one aims to avoid having to retrain the model repeatedly.

Even though performance is known to be affected by the perturbations characteristic of dynamic environments, their effect on fairness is still largely unexplored. Our goal is to make progress in this direction, studying robustness and observational fairness in dynamic environments. A significant part of our work will focus, not only on how perturbations affect the performance of ML models but also on its impact on fairness.

The datasets we build our work on are based on a real-world fraud detection use case and a common dataset used in the *fair ML* literature, where fairness concerns are present. We apply different perturbations to these datasets and study their impact on the model's fairness and performance. It is also worth noting that these datasets feature different configurations regarding bias and drift, making them a good fit for our work's setup.

1.1 Contributions

The main contributions of this thesis are briefly described in the following paragraphs. The first contribution consists of the proposition of a perturbation taxonomy that outlines common problems that affect ML models after they are deployed in the real world. When testing the model's robustness against perturbations, the literature mostly focuses on adversarial attacks. In our selection, we consider the most common obstacles models face in the fraud detection scenario, extending our taxonomy beyond this sphere. Besides the adversarial attacks, we consider data issues, that refer to problems that harm the quality of data (like missing values, for instance).

Given the proposed perturbation, we suggest a new view on robustness. Our view considers robustness as context-dependent, and, with this in mind, we propose a framework to evaluate robustness against each perturbation in the taxonomy as the second contribution. Additionally, we include fairness concerns in our approach.

The third contribution consists of a study regarding the robustness of popular ML models. In order to do so, we employ our robustness framework, evaluating the fairness and performance of models in stressful environmental conditions - the perturbations we proposed.

Lastly, the fourth contribution consists of using a common method for enhancing robustness in the literature and then recurring to our proposed framework to compare it to a model that was developed with no robustness concerns. Our goal here is to show the value of having a tool to evaluate robustness in the process of choosing the best model to deploy.

In summary, the main contributions of this dissertation are:

- A perturbation taxonomy broader than just adversarial attacks.
- A novel framework for evaluating robustness, considering not only performance but also fairness, and aligning with the perturbation taxonomy.
- Employing our framework to assess the robustness of popular ML methods against perturbations.
- Comparing two models developed with and without robustness concerns and highlighting their differences resorting to our framework.

1.2 Outline

This document is organized as follows:

- **Chapter 2:** Background and State of the Art - A background on the current fair ML literature, as well as a background on the literature on dynamics environments problem detection and mitigation approaches. Finally, we present the approaches taken concerning both fairness and robustness in dynamic systems together.
- **Chapter 3:** A Framework for Evaluating Robustness - This chapter presents a holistic view of the lifecycle of ML models and how our setup aims to replicate it. Furthermore, it features the propositions of the perturbation taxonomy and a robustness evaluation framework for assessing model resilience against these perturbations.
- **Chapter 4:** Experimental Setup - A view on the datasets and model development used for this research.
- **Chapter 5:** Robustness Against Data Issues - Evaluation of models' robustness against data issues on different datasets.
- **Chapter 6:** Adversarial Robustness - Similarly to the previous chapter, consists of evaluating the model's robustness against a perturbation, in this case, adversarial attacks.
- **Chapter 7:** Enhancing Model Robustness - This chapter features the development of a robust model using a common method in the literature and then the comparison against its robustness blind counterpart.
- **Chapter 8:** Conclusion and Future Work - The final chapter consists of a recap of the work conducted in this thesis and what might follow it.

Chapter 2

Background & State of the Art

This section presents the current literature on approaches to deal with the problems mentioned: fairness and robustness in dynamic environments. Following a brief introduction to ML and classification, we present an overview of observational fairness followed by a review of numerous studies in the field. We discuss the metrics used to represent fairness, the inherent trade-offs among these metrics, and between them and performance. Finally, we review the literature on the fairness-enhancing mechanisms. Afterward, we move to robustness and the impact of dynamic environments on ML systems, enumerating existing problems and the literature on approaches regarding them. This chapter closes by reviewing studies that relate robustness and observational fairness together in dynamic environments.

2.1 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence dedicated to finding statistical relationships in data to solve problems. Usually, ML models learn patterns from a dataset during a training phase and then apply what they've learned in unseen data after being deployed, in *production.*

Supervised learning is one of the main paradigms of ML, where a model's goal is to predict an unknown outcome y given a set of features x. During training time, the model has access to a dataset composed of input-output pairs, where the idea is to approximate the ”true” decision function $y = f(x)$ by fitting a hypothesis $h(x)$ that best maps any given input to the correct output.

Thanks to its configuration, supervised learning is growing in use [10]. Practitioners collect historical data on both the features x and the label y so it can be used at training time. During testing, the model has only access to an input and is asked to predict the output. A model is said to perform if it is able to correctly make predictions of the output given unseen data.

2.1.1 Classification and Evaluation Metrics

In classification, the output is a category or a set of categories (e.g. a model that predicts if an image is a cat or dog), unlike in regression, where the output is continuous (e.g., a model that predicts the expected returns in an investment). In our setting, that is financial fraud detection, a model's task is to predict whether an instance is legitimate (class 0) or fraudulent (class 1), making it a binary classification setting.

Usually, there are two modules in a classifier: one learns the conditional distribution of the target in the form of a vector of conditional probabilities, also named scores; the other one takes this vector and gives an integer as a prediction, representing the predicted class. To complete this thresholding dynamic, is essential to take classification metrics into account.

In binary classification, the key evaluation metrics are summarized by the confusion matrix, shown in Figure 2.1.

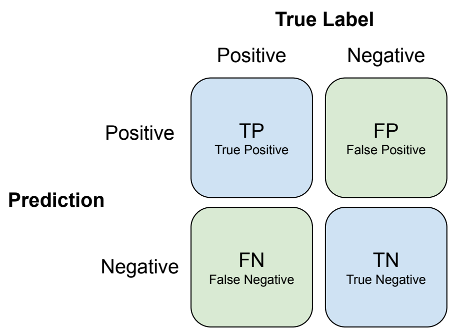


Figure 2.1: A confusion matrix evaluates predictions in a binary classification problem.

In our scenario, a fraudulent instance has a positive true label. This means that, in case the model is able to detect the fraud, it will lead to a true positive. If the model is not able to identify it as a fraudulent instance, it will result in a false negative. The same goes for legitimate instances, regarding true negatives and false positives. These four values are the base of most classification metrics.

An additional metric for evaluating the performance of a binary classifier, without the need for choosing a threshold, is the AUC, which consists of the area under the receiver operating characteristic (ROC) curve. The ROC curve represents every pair of (FPR, TPR) at each possible decision threshold. The AUC is the area under that curve, and, if the model is better than random, its value is higher than 0.5. However, if TPR=FPR for every threshold, it will lead to AUC=0.5, and that model is not able to distinguish between classes, thus behaving as a random classifier.

By providing valuable insights regarding the TPR-FPR trade-offs for different thresholds, it becomes a useful tool for choosing the best value for the threshold. This decision depends on the task at hand, and practitioners proceed according to their objectives.

2.2 Observational Fairness

2.2.1 Overview

Fairness refers to how impartial a system is. Intuitively, fair decisions cannot be based on special protected categories of personal data, e.g., sensitive attributes such as gender or race. However, there is no universally accepted mathematical definition for measuring fairness. Simply removing the sensitive attributes is not the optimal solution, because other features can serve as a *proxy* for said attribute (as they are correlated with them), and may even lead to a decrease in both performance and fairness [11]. To better understand algorithmic unfairness, we need to understand the different metrics used for measuring unfairness and their relationship to distinct principles of fairness. In this work, we will focus on the observational fairness metrics outlined by Barocas et al. [11]. Before discussing the fairness metrics, we introduce the notation:

- **X** - the features in the dataset, besides the protected attribute(s).
- **A** - protected attribute(s).
- **Y** - target variable.
- **R** - score attributed by the classifier
- **Ŷ** - binarized outcome of the classifier

Among the definitions proposed by Barocas et al [11], we will focus on observational fairness, where the criteria are properties of the joint distribution of the features X, the score R, the sensitive attribute A, and the target variable Y.

Observational fairness criteria can be divided into two categories: group and individual fairness. Group fairness measures if the outcomes are fair regarding the aggregate of results for protected groups. A protected group is a set of data observations with the same sensitive feature A for which we want to guarantee fair outcomes. Said feature A may or may not be available when training the model. On the other hand, individual fairness has as a goal to guarantee fairness among single instances of the dataset, meaning that similar individuals should have the same outcome. In our work, we opt to use the group fairness criteria, as individual fairness depends on a measure of similarity between individuals, which can become a source of bias itself.

2.2.2 Definition and Metrics

Broadly speaking, all fairness metrics equalize some group-dependent statistical quantity across groups characterized by some sensitive feature A. In the literature, there are numerous criteria, each one trying to formalize different concepts of fairness. Simplifying the notion of fairness, criteria can be divided into three fundamental groups, each trying to equalize a different statistical quantity:

- Acceptance rate - $\mathbb{P}(\hat{Y} = 1)$;
- Error rates - $\mathbb{P}(\hat{Y} = 0|Y = 1)$ and $\mathbb{P}(\hat{Y} = 1|Y = 0)$ of a classifier $\hat{Y}$;
- Outcome frequency given score value $\mathbb{P}(Y = 1|R = r)$ of a score R.

These three groups can be further explained by simple conditional independence statements. Below we introduce each in more detail.

Principle 1. Independence: this first criterion requires the sensitive attribute to be statistically independent of the score:

$$R \perp\!\!\!\perp A.$$

If R is a score function that satisfies independence, then every classifier of the form $\hat{Y} = 1(R > t)$ also satisfies independence for any threshold *t*. This is true as long as the threshold is not group-dependent, whereby group-specific thresholds may not preserve independence.

Pertinent metrics of independence, when applied to a classifier $\hat{Y}$, are *demographic parity* and *disparate impact*.

Metric 1. Demographic Parity [12, 13]: this criterion is satisfied if the acceptance rates are the same for each sensitive group

$$\mathbb{P}(\hat{Y} = 1|A = a) = \mathbb{P}(\hat{Y} = 1|A = b), \tag{2.1}$$

for all groups *a* and *b*. A relaxation of the constraint allows the acceptance rates to be slightly different, using a constant of slack $\epsilon > 0$.

Metric 2. Dispare Impact [14]: this is an adaptation of demographic parity to quantify the metric and to confer flexibility to the results. The **p%-rule** [15], that determines that the ratio between the minimum and the maximum registered values for a metric should not be lower than a given pre-determined value *p*, is applied to demographic parity. The resulting metric is

$$DI = \min\left(\frac{\mathbb{P}(\hat{Y} = 1|A = a)}{\mathbb{P}(\hat{Y} = 1|A = b)}, \frac{\mathbb{P}(\hat{Y} = 1|A = b)}{\mathbb{P}(\hat{Y} = 1|A = a)}\right). \tag{2.2}$$

By definition, we compare this metric to *p*, so that $DI \geq p$. A commonly applied value for p is 80%.

Limitations of Independence. Independence-based criteria have been criticized due to their disregard for the true value of the target variable [16, 17], leading to less efficient systems and ignoring meritocracy [18]. This means that ill-intentioned decision-makers can take advantage of this metric. A good example is the *glass cliff*, where women tend to be named CEO more often when a company is struggling [19].

Principle 2. Separation: this next principle addresses the complaints regarding the previous one, as there is a difference between accepting a positive and a negative instance. By using the target variable Y, we divide the data into strata of the equal claim to acceptance, giving us a sense of *merit*. To satisfy these considerations, we need to guarantee conditional independence within each of these strata

$$R \perp\!\!\!\perp A \,|\, Y. \tag{2.3}$$

We can display conditional independence by a graphical model, as shown in Figure 2.2 a).

Metric 3. Equalized odds [16]: this metric is satisfied if both the False Positive Rate (FPR) and the True Positive Rate (TPR) are the same across all groups *a* and *b*

$$\mathbb{P}(\hat{Y} = 1|Y = 0, A = a) = \mathbb{P}(\hat{Y} = 1|Y = 0, A = b),$$

$$\mathbb{P}(\hat{Y} = 1 | Y = 1, A = a) = \mathbb{P}(\hat{Y} = 1 | Y = 1, A = b). \tag{2.4}$$

Metric 4. Predictive Equality [20, 21]: this represents a relaxation of equalized odds, discarding the TPR and using only the FPR:

$$\mathbb{P}(\hat{Y} = 1 | Y = 0, A = a) = \mathbb{P}(\hat{Y} = 1 | Y = 0, A = b). \tag{2.5}$$

Metric 5. Equal Opportunity[16]: just as predictive equality, it represents a relaxation of equalized odds. However, equal opportunity goes the other way around, discarding the FPR and focusing only on the TPR balance

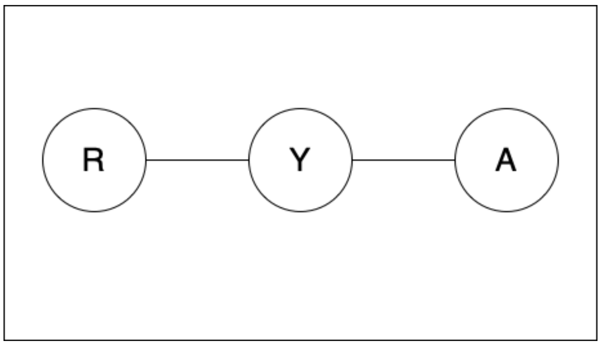
$$\mathbb{P}(\hat{Y} = 1 | Y = 1, A = a) = \mathbb{P}(\hat{Y} = 1 | Y = 1, A = b). \tag{2.6}$$

Limitations of Separation. Comparing error rates across sensitive groups has been subject to criticism. The first argument against these criteria states the fact that an optimal classifier does not need to have the same error rates for different groups [11]. Moreover, if the prevalence of different groups is different, an optimal classifier will have different error rates. Enforcing equal error rates may lead to a classifier that performs worse for a group than it could, thus making its fairness questionable. Another concern regarding these criteria questions the use of a target variable as a stand-in for *merit*. These target variables often have biases embedded within, so it raises doubts if comparing error rates would be the correct approach. Furthermore, this issue could be generalized to supervised learning as a whole [11].

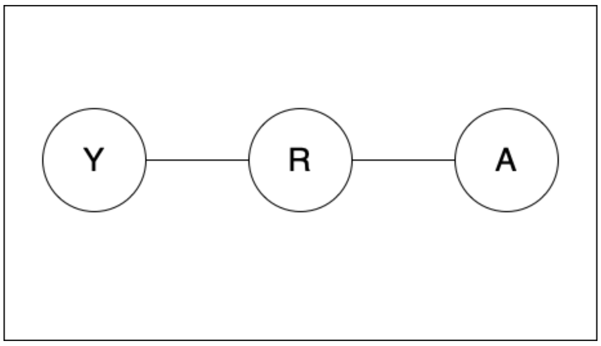
Principle 3. Sufficiency: The third and final condition is sufficiency, which assumes that the score R already takes into account the sensitive feature A to predict the target variable:

$$Y \perp\!\!\!\perp A \,|\, R. \tag{2.7}$$

We can once again display this relationship in a graphical display (Figure 2.2 b)).



(a) Separation



(b) Sufficiency

Figure 2.2: Fairness principles representation.

The idea of sufficiency is that the model is independently calibrated for each group, thus the score of the model for any instance reflects the probability of it being a positive instance.

Metric 6. Group Calibration [21]: this metric implies sufficiency, and both are closely related. It proves its value when model scores are not intended to be thresholded or the threshold changes frequently.

$$\mathbb{P}(\hat{Y} = 1 | R = r, A = a) = r. \tag{2.8}$$

More often than not, unconstrained supervised learning leads to sufficiency as it implies calibration by group [11]. However, for this to be possible, the model must be able to detect group membership.

Limitations of Sufficiency. Even though sufficiency usually comes for free due to common ML practices, imposing this constraint usually is not much of a fairness intervention, as it would not change much in current practices.

All things considered, group fairness metrics find their usefulness in their ability to capture the impact of predictions in different groups. By having access to true labels, we can compare error rates among groups and evaluate how models perform in each of them. This information proves itself valuable when in high-stakes decisions, as fairness must be accounted for. Anyhow, one should always decide on the most appropriate metric for different tasks, as all of them have inherent trade-offs and are based on different fairness principles.

To conclude the topic of fairness definitions and metrics, we encapsulate the main concepts in a taxonomy diagram, visible in Figure 2.3.

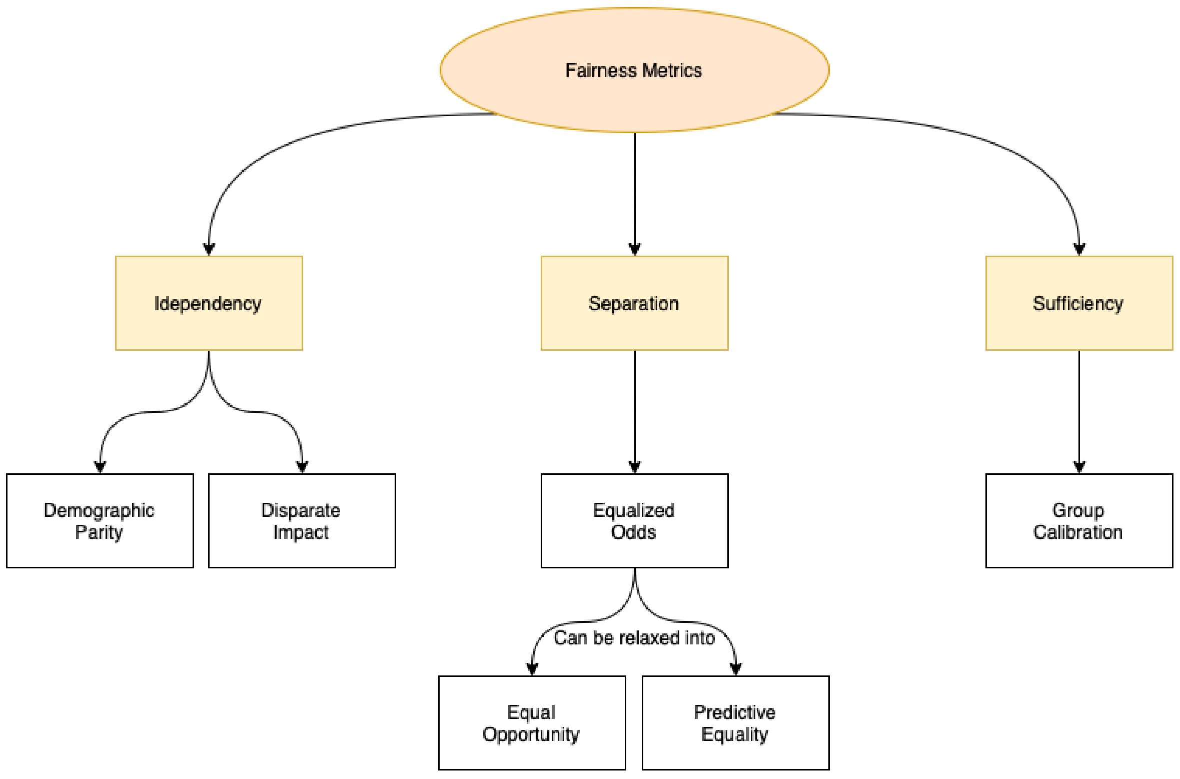


Figure 2.3: Taxonomy of Observational Group Fairness Metrics.

2.2.3 Inherent Trade-Offs in Observational Fairness

As demonstrated earlier, each situation calls for a different approach regarding fairness metrics. Each metric has advantages and disadvantages, so each has a different setting where it should be used. We now present the inherent trade-offs between metrics themselves and between them and accuracy.

Fairness Measures Trade-offs

By using the previously mentioned criteria, we constrain the joint distribution in non-trivial ways. This implies that using more than one criterion together will be so restrictive that the only solutions available are trivial. Various studies have shown that it is impossible to simultaneously satisfy multiple notions of fairness. For example, it has been shown that if a classifier score is calibrated and group-wise prevalences are different, there is no way to balance both FPR and FNR [21]. Corbett-Davies and Goel [22] dive deeper into these trade-offs and discuss their statistical limitations.

The definitions based on the target variable, such as equalized odds, and the model decision (sufficiency) are usually mutually exclusive. The reason for this is that when the TPR and FPR are fixed in a binary decision scenario, the values for precision are solely determined by the prevalence of the target variable in test [11, 21, 23]. This relation can be expressed by [21]:

$$FPR = \frac{p}{1-p}\frac{1-PPV}{PPV}TPR, \tag{2.9}$$

with p representing the target variable prevalence, PPV the precision, and TPR the true positive rate (recall). To achieve parity in every single one of the four metrics, it is necessary that all the groups have the same target variable prevalence or the classifier to be perfect [23].

While fairness is a product of model and data interactions, prevalence is only related to the data itself. One should at least study group-wise prevalence to make more informed decisions when choosing which metric to use. That said, one can conclude that the definition of fairness to adopt is highly dependent on the task at hand and its constraints. A decision chart [24] was proposed to help choose the metric or fairness definition that better suits their needs and knowledge.

Fairness-Accuracy Trade-offs

The literature focuses heavily on the trade-off between fairness and accuracy. A theoretical analysis is conducted by Corbett-Davies et al. [20]. Regarding this trade-off, one of the main goals of FairML is to design fairness-aware algorithms that allow for higher fairness by compromising accuracy as little as possible. Intuitively, we may consider adding fairness constraints to a model as a constrained optimization problem. In this scenario, it is clear that adding these constraints will lead to similar or worse accuracy than in a model where maximizing accuracy is the sole objective. As FairML focuses on limiting this performance decrease, a broad scale of benchmarks has been developed, comparing various approaches for fairness-enhancing mechanisms [25, 26]. These studies provide a good understanding of how the different methods proposed compare in similar scenarios.

It is also important to note that even though the usual case regarding performance decrease is the one referred to previously, there are also exceptions. For example, if our metric of interest is equalized odds, a perfect classifier also achieves a perfect value for fairness.

2.2.4 Fairness Enhancing Mechanisms

The literature covers many approaches when it comes to enhancing fairness in ML. These approaches are usually divided into three different categories: pre-processing, in-processing, and post-processing [27]. In this section, we address these different approaches and enumerate various methods proposed to combat bias and unfairness.

Pre-Processing

Pre-processing methods recognize that data is often the cause of unfairness. Therefore, pre-processing approaches often alter the training dataset before training, thus providing the downstream ML pipeline with unbiased data. Methods in this category are arguably the most flexible, as they work directly with training data allowing for the use of different configurations of subsequent models. Commonly used techniques include changing the labels of some instances or reweighting them to make the classifier fairer [28] and modifying feature representation so that the classifier obtained is fairer [14, 29].

Generally, the labels that are changed are those of samples that are closer to the decision boundary. Label massaging is an example of this idea [28]. A ranker can be used to select the best candidates for relabeling. Regarding reweighing, instead of changing the labels, one can assign weights to the instances that constitute the dataset. The idea is to carefully choose the weights with regard to the sensitive attribute without having to change the labels, for example by using frequency counts. In the context of fraud detection, one might assign a higher weight to non-fraudulent observations from the most fraudulent group, leading the model to avoid focusing on the fact that said group is usually more fraudulent. Another possible approach is sampling, this means calculating the sample size for all combinations of the sensitive attribute and class value to make the dataset less biased.

On the other hand, modifying feature representation, or fair representation learning, aims to obtain a new data distribution, where it is no longer possible to find the relationship between the sensitive attribute, the features, and the target variable, but the remaining relevant information for the task at hand is preserved. An example of an approach following this principle is the work by Xu et al. [29], which consists of developing a generative adversarial network (GAN) for fair ML. Their proposal consists of a sample generator, a classifier, and three discriminators (one to predict if a sample is drawn from a real distribution or is generated, one to predict the sensitive attribute, and a third one to distinguish between values predicted by the classifier) to assist in adversarial learning. To achieve fair data generation and classification, the generating model is co-trained with the classifier in joint adversarial games with the discriminators. A different approach [14] proposes modifying features in the dataset to make the distribution of privileged and unprivileged groups similar: this work even suggests a tuning parameter that controls the trade-off between fairness and accuracy. The principle on which most fair representation works are based is independence, meaning that it often presents guarantees for group demographic parity, but not other, possibly more pertinent, fairness metrics.

In-Processing

In-processing methods consider that the learning process often becomes biased by dominant features (or other distributional effects). Common approaches incorporate fairness constraints in the objectives of the model, making it converge to the desired accuracy and fairness values. These approaches can be used even when the data has bias embedded within.

Examples include the proposal of Kamishima et al. [30] proposal to add a regularization term to the objective function, penalizing the mutual information between the sensitive attribute and the predictions. This method also provides a tuning parameter to deal with the fairness-accuracy trade-off. Zafar et al [13, 31] suggest adding constraints to the classifier that require satisfying either *disparate impact* [31] or *equalized odds* [13]. Kamiran et al. [28] study an alternative approach in which the fairness constraint is pushed deeper into a decision tree model by adjusting its splitting criterion to maximize information gain between the split attribute and the class label and minimizing information regarding the sensitive attribute. Another approach consists of training separate models for each group in the sensitive attribute, and then directing the input to the correct classifier, as proposed by Calders and Verwer [32].

Even though pre-processing methods offer flexibility, in-processing approaches are usually more versatile, as they can be designed to impose different fairness definitions and metrics. Furthermore, as they directly use the fairness metrics in the design, it is expected that the fairness concerns are more likely to be satisfied.

Post-Processing

Post-processing techniques assume that the output of the learned model may be unfair to some protected groups. Assuming this, these approaches apply transformations to the model output to improve prediction fairness. One example of a post-processing approach is the work by Hardt et al.[16], which presents a method for satisfying fairness metrics by simply tuning the classification threshold. However, instead of using a global threshold, by navigating the ROC space, the study aims to find group-specific thresholds that lead to the satisfaction of the required fairness metric, *equalized odds* or *equal opportunity*. Corbett-Davies et al. [20] conduct a similar study but regarding *demographic parity*. The latter approach is easier to explain as you simply need to decrease the threshold for the groups with fewer positive instances predicted. Another approach is to use a decoupling technique to learn a different classifier for each group as proposed by Dwork et al. [33]. Moreover, the authors combine a *transfer learning* technique with their procedure to learn out-of-group samples.

These methods are also very flexible as they only need to access the prediction and the sensitive attribute information, without access to the model implementation. Their usefulness is found in high-stakes decision-making settings, where there is no access to the data before training and it is impossible to change the optimization strategies.

2.2.5 Extension to Dynamic Environments

The works up to this point study fairness in static environments. Moreover, the majority of the work in FairML focuses on these scenarios. However, most real-work applications feature dynamic environments, where the settings at training time are not the same at deployment time. In these scenarios, both the data and its interactions with the model feature unpredictable changes, ranging from feedback loops between the model and the environment to shifts caused by the environment itself.

To better understand the effects of these dynamics on fairness, it was first necessary to get a grasp of the concepts of data bias and its interactions with the models. In the remainder of this chapter, we will transition to dynamic environments, studying their impact on both accuracy and fairness.

2.3 Robustness and Dynamic Environments

Unlike static environments, dynamic ones feature complex relations among their elements which influence model behavior. This means that ML models deployed in those settings need to withstand the effects of such dynamics. To address this topic, we will provide a taxonomy of situations that pose obstacles to the models. Furthermore, we will review some of the approaches that have been proposed to mitigate their impact and discuss the role of robustness in this problem.

2.3.1 Robustness

A common approach to dealing with the undesired dynamics of the environment is to constantly retrain the model with more recent data. However, this process is costly and sometimes impossible, as in cases such as the fraud detection scenario, there is an inherent delay in the labeling process (often, a fraudulent instance takes a long time to be uncovered, leading to labeling uncertainty). To avoid this struggle, one should aim to build robust models. Robustness can be defined as the resilience of the system's correctness in the presence of perturbations [34]. The causes of the perturbations can be malicious intent (i.e., adversarial attacks) or of other nature (i.e., covariate shift due to consumer habit change). Regarding the correctness of the model, it can refer to various properties of the model, namely performance metrics and fairness. All in all, it means that robustness is tied to the ability of a model to maintain similar levels of correctness with and without the presence of invalid inputs or stressful environmental conditions.

Instead of focusing on evaluation, the field of robustness is heavily oriented toward the development and optimization of methods for adversarial attack generation. This leads to the small number of available metrics to measure robustness. Some of the available ones focus on local robustness [35], which takes into account a single or a small group of input instances and sets a distance in feature space where no adversarial attack should be possible, or in global robustness, which is representative of the whole set of instances available but is harder to reach [36]. Another definition of robustness is that of probabilistic robustness [37]. In this case, instead of relying on the basis of adversarial attacks to estimate robustness, it is measured globally, in the neighboring data points at a given distance for each instance.

2.3.2 Distribution Shift

A distribution shift is a phenomenon that implies a change in the distribution of the data between training time and testing time. Among the distribution shifts studied in the literature, the two major categories addressed are *covariate shift* and *concept shift*.

Covariate Shift

Covariate shift refers to the distribution of the input features, or covariates, not being the same at training and testing time, but the same not happening to the conditional distribution of the outcomes given the features is:

$$\mathbb{P}_{pre}(A, X) \neq \mathbb{P}_{post}(A, X) \; and \; \mathbb{P}_{pre}(Y|A, X) = \mathbb{P}_{post}(Y|A, X). \tag{2.10}$$

This can lead to poor performance, as the model was not trained in the deployment distribution. There are a variety of reasons for *covariate shift* to happen, namely changes in data collection over time and differences in the populations used for training and testing (i.e., a model trained before a pandemic and deployed after).

To address *covariate shift*, the literature proposes methods for its detection and mitigation. Shimodaira [38] suggests alleviating the influence of *covariate shift* by weighing the log-likelihood function according to *importance* (the ratio between testing and training input densities). Regarding model selection, Sugiyama et al. [39] studies a new cross-validation variant called importance-weighted cross-validation, where the validation error is weighted according to *importance.* Another approach is reweighing by kernel mean matching, proposed by Gretton et al. [40]. In this method, the authors account for the difference between distributions by reweighting the training points in a way that the means of the training and test points in a reproducing kernel Hilbert space are close.

Concept Shift

Conversely, *concept shift* is the scenario where the distribution of the input features stays the same, but the conditional distribution of the outcomes given the features changes from training time to testing time:

$$\mathbb{P}_{pre}(A, X) = \mathbb{P}_{post}(A, X) \; and \; \mathbb{P}_{pre}(Y|A, X) \neq \mathbb{P}_{post}(Y|A, X) \tag{2.11}$$

As expected, this phenomenon can also lead to a performance decrease. A reason that can lead to *concept shift* is applying the same model to different populations from the one it was trained on, for example in a different geographical location. The associated change in context leads to a different mapping from features to outcomes.

Most works on *concept shift* focus on its detection. The approach offered by Webb et al. [41] is a *concept drift mapping* that provides a quantitative description of drift and shift in marginal distributions paired with visualization tools. Vorburger and Bernstein [42] propose an entropy measure that reveals differences between older and more recent instances, adapting the window size of their control model when its value drops below a given threshold; at this time, it forgets the current model and learns a new one.

Dynamic Bias

A different way to look at the problem of distribution shifts is to consider the bias conditions change over time. Data bias is a central topic in our work, so it is important to state these shifts can also be associated with dynamic bias conditions. For example, if a model is trained on a predominantly male dataset, it may develop biases related to men. If the model is then exposed to a female-dominated dataset, its biases might shift towards women.

2.3.3 Performative Prediction

In decision-making scenarios, using a model to support the decision process, can trigger actions that influence the outcome it aims to predict. This phenomenon is called *performative prediction* and it is associated with a distribution change in the target data caused by the model itself. To better understand this dynamic, one can think of a traffic predictor, which can change traffic patterns if it has a significant number of users.

Performative prediction is a particular case of *concept shift*, thus affecting the mapping from the features to the outcomes. However, studying this problem at an abstract level has led to several difficulties, namely creating a unified language and objective. Perdomo et al. [43] address this problem, proposing a risk minimization framework, featuring an iterative process named *repeated risk minimization* that aims to converge to a minimal loss in the distribution, a situation the authors named *performative stability*.

Strategic Classification

One special case of performative prediction is *strategic classification*. Knowing information about the classifier, individuals may intentionally set their attributes in a configuration that leads to favorable classification - a process called *gaming*. In fraud detection scenarios, it represents a common strategy used by fraudsters. This process usually leads to a performance decrease.

Strategic classification aims to anticipate this behavior. It is an adversarial setting, where an institution deploys a classifier and users manipulate their attributes to achieve desired outcomes. Hardt et al. [44] propose a general model based on a two-player game between a party that wishes to learn a classifier and a party that is being classified. Furthermore, Perdomo et al. [43] also extend their work to strategic classification, showing that the optimal strategy for this task coincides with their definition of *performative optimality*.

Selective Noisy Labels

In fraud detection, rejecting one instance of an account opening, due to believing it is fraudulent, means that its true label will never be observed. The reasoning behind it is that if the account opening is not allowed, there is no way to know if that account would actually be fraudulent in case it would have been opened. Consequently, due to the importance of retraining with recent data in dynamic environments, this situation becomes an obstacle, as the system fully determines the outcome of these instances. This makes this scenario *performative*, and it is an example of the selective noisy labels problem. We can define this problem as:

$$\mathbb{P}^*(Y|A,X) \neq \mathbb{P}(Y|A,X), \tag{2.12}$$

where $\mathbb{P}^*$ is the observed distribution and $\mathbb{P}$ is the ground-truth distribution. This means that some instances will be incorrectly labeled by the model. Assuming these labels to be truthful has an impact on the performance of models, as most metrics are based on ground-truth labels. There have been works focusing on this problem, namely the work by Natajaran et al. [45], which proposes two methods to suitably modify any given surrogate loss function.

2.3.4 Adversarial Attacks

Adversarial attacks are situations where an attacker intentionally feeds the model with misleading or incorrect data, thus causing the model to make mistakes or behave in an unwanted manner. For example, a model that is trained to predict if a customer will default on a loan based on credit history and personal information might be attacked by an ill-intentioned user that manipulates the data, making the model predict that a particular customer will not default, even if that is not the case.

There are multiple *attack surfaces* that can be targeted. An adversary can attempt to manipulate either the *collection* or the *processing* of data to disrupt the model. The most common attacks are *evasion* attacks, where the adversary tries to evade the system by adjusting malicious samples during the testing phase. Another attack type is *poisoning*, where the main goal is to contaminate the training data, by crafting samples that compromise the whole learning process. An approach that is also taken by attackers is to try to learn as much as possible about the learning algorithm, given black-box access to the model. The latter is called *exploratory attacks*. [46]

Focusing on the most common attack type, the *evasion attacks*, we can also group attacks into different categories according to their *testing phase capabilities*. Testing phase attacks can be broadly classified as either *white-box* or *black-box* attacks [46]. This categorization derives from the level of knowledge the attacker has regarding the model.

White-Box Attacks

White-box attacks assume adversaries have total knowledge about the model used for classification: the algorithm used in training, the training data distribution, and the parameters of the fully trained model architecture. In addition, attackers can also perform infinite queries and try out as many attacks as desired, naturally resulting in more serious threats to a target model.

Goodfellow et al. [47] were the first to show that it is possible to fool linear models with adversarial samples as long as their input has sufficient dimensionality. The authors propose the fast gradient sign method (FSGM) [47], which uses the differentiable properties of a given network to find the smallest perturbation that results in a change of the output by maximizing the loss. Projected gradient descent [48] was proposed to develop stronger attacks based on FSGM. This attack applies FSGM iteratively, updating the adversarial samples using gradient descent [49] upon the negative loss function while constraining the perturbations to be within a predefined epsilon-bound. Another popular attack in the literature is DeepFool [50], in which the general idea is to iteratively linearize classifiers around the current iteration point and then find the minimal perturbation that leads to misclassification. A more recent approach is the C&W attacks [7]; this formulation simply tries to find the smallest perturbation vector that if added to the sample leads to an output change. The authors use l_0, l_2, and l_∞ to measure the size of the perturbations.

Black-Box Attacks

On the other hand, *black-box* attacks assume the attackers have no information about the model, making them less threatening than white-box attacks, but more realistic. More often than not, attackers only have access to the hard label produced by the model and have a limited number of queries they can make before being denied access. These attacks can also be characterized by the kind of access they have to the model output, as some attacks assume access to the output probabilities for each class given by the model and others only to the hard labels. The latter is the most realistic setting of all, and the one that resembles financial fraud detection scenarios.

Some attacks in the literature are based on the zeroth order optimization, an optimization method that does not rely on gradients, which proves itself useful in problems that are convex (like settings when the attacker only has access to the hard label). The attack proposed by Cheng et al. [51] proposes a function $g_x(\theta)$ that returns the distance from an instance x and its nearest adversarial sample along the direction θ. The authors use a gradient approximation to iteratively update the search direction θ while we are searching for a misclassification. When it comes to distortion, these attacks are comparable to the white-box C&W attacks [7]. Later, this method was evolved into SignOPT [52], which assumes more relaxed approaches to directional derivatives, leading to higher success rates for the same number of queries.

Another approach consists is walking across the boundary in search of the optimal adversarial example. Some examples of these attacks are the boundary attack [53] and the HopSkipJump attack [54], both of which find the boundary through binary search. Then, the former takes random walks across the decision boundary in order to find the closest adversarial sample to the original sample; the latter uses a gradient-based approach in the boundary to get closer to the original sample. More recently, another attack was proposed, the RayS Attack [55], that uses the l_∞ norm to measure perturbation size and limits the search of the perturbation direction to a limited set.

Adversarial Attacks Against Tabular Data

The attacks explained thus far are developed with image data as its primary target. Tabular data poses a different challenge, as there are dense numerical features and sparse categorical features [56], not all features are accessible to attackers, and not every feature is as important when it comes to attack detection. In financial fraud detection, tabular data is the most common type of data used, giving studies on this domain an increased importance for our work. Cartella et al. [57] adapt three previously mentioned attacks meant for image data to tabular data, assessing their effectiveness.

The first one is the Boundary Attack [53]. This attack starts by finding a random sample that is adversarial (the output of the model is different for this sample and the original sample). The initial sample is then used in a binary search, as well as the original sample, in order to find the decision boundary. It is at this decision boundary that a random direction is drawn from an i.i.d. Gaussian. The adversarial sample is then moved in that direction, followed by a minor adjustment back toward the original image. The step sizes are dynamically adapted throughout this process, as shown in Figure 2.4. This process is repeated for a chosen number of iterations, returning the best possible adversarial sample (smallest perturbation) according to a given norm.

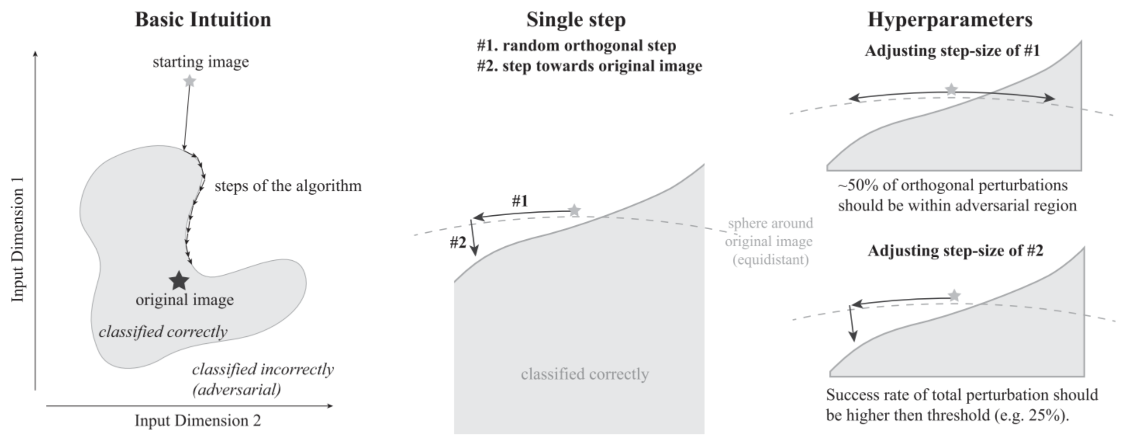

Basic Intuition
starting image
steps of the algorithm
Input Dimension 1
original image
classified correctly
classified incorrectly (adversarial)
Input Dimension 2
Single step
#1. random orthogonal step
#2. step towards original image
#1
#2
sphere around original image (equidistant)
classified correctly
Hyperparameters
Adjusting step-size of #1
~50% of orthogonal perturbations should be within adversarial region
Adjusting step-size of #2
Success rate of total perturbation should be higher then threshold (e.g. 25%).

Figure 2.4: (Left) The Boundary Attack performs rejection sampling along the boundary between adversarial and non-adversarial images. (Center) In each step, we draw a new random direction by (1) drawing from an iid Gaussian and projecting on a sphere, and by (2) making a small move towards the target image. (Right) The two-step sizes (orthogonal and towards the original input) are dynamically adjusted according to the local geometry of the boundary. [53]

The second attack is the HopSkipJump Attack [54]. This attack differs from the previous one in the optimization approach. Instead of sampling random directions and using them to find a smaller perturbation, this method uses a gradient-based approach. The gradient is computed at the boundary using a convex function defined by the authors in equation 2.13.

$$S_{x^*}(x') := max\, F_c(x') - F_{c^*}(x') \tag{2.13}$$

Focusing on a binary classification, attacks are untargeted (do not target any specific class, simply want the model to predict a different class than the original one). Knowing this, x^* represents the targetted input, and the goal of the attack is to change the original classifier decision from class $c^* := C(x^*)$ to the other class, $c \neq c^*$. Finally, F_c corresponds to the output probability distribution over class c.

The gradient is then computed for an iteration $x_t \in$ bd(S_{x^*}), where

$$bd(S_{x^*}) = \{z \in [0,1]^d \mid S_{x^*}(z) = 0\} \tag{2.14}$$

with d being the number of features. Finally, the gradient is estimated using a Monte Carlo estimate that is valid if the input sample is sufficiently close to the boundary.

$$\widetilde{\nabla S}(x_t, \delta) := \frac{1}{B}\sum_{b=1}^{B} \phi_{x^*}(x_t + \delta u_b) u_b \tag{2.15}$$

where $\{u_b\}_b^B = 1$ are iid draws from the uniform distribution over the d-dimensional sphere, and δ is a small positive parameter. Also, $\phi_{x^*}(x') := sign(S_{x^*}(x'))$. After computing the gradient, the next iteration is computed through a geometric progression and binary search

$$x_{t+1} = \alpha_t x^* + (1-\alpha_t)\left\{x_t + \epsilon_t \frac{\nabla S_{x^*}(x_t)}{\|\nabla S_{x^*}(x_t)\|}\right\}, \tag{2.16}$$

where $\alpha_t \in [0,1]$ is chosen so that $S_{x^*}(x_{t+1}) = 0$, as the gradient-direction estimate is only valid near the boundary. This process is repeated for the chosen number of iterations (the process is observable in Figure 2.5).

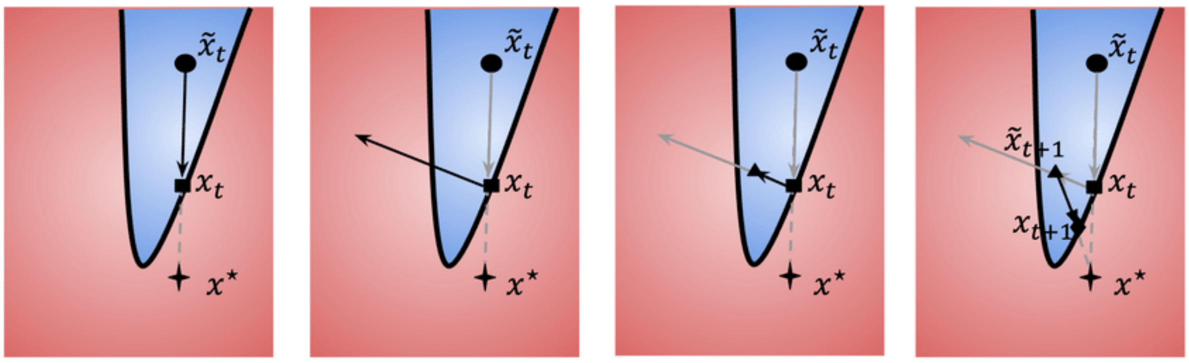


Figure 2.5: Intuitive explanation of HopSkipJumpAttack. (a) Perform a binary search to find the boundary, and then update $\tilde{x}_t \rightarrow x_t$. (b) Estimate the gradient at the boundary point x_t. (c) Geometric progression and then update $x_t \rightarrow \tilde{x}_{t+1}$. (d) Perform a binary search, and then update $\tilde{x}_{t+1} \rightarrow x_{t+1}$ [54].

The third and final attack is a Zeroth Order Optimization attack, however, as it has access to the probabilities outputted by the model instead of only to the hard labels, makes it less appealing in our setting.

The contribution of the authors consists of the adaptation of these attacks to tabular data, applying custom norms, editability constraints, and enforcing realist values.

Adversarial Robustness

The most common approach to deal with this problem is using *adversarial training*. Generally speaking, adversarial training consists of a data augmentation technique that generates new adversarial samples from a set of victim examples. This method focuses on exposing flaws in the classifier's way of modeling its decision function. So, more adversarial examples are iteratively generated when the model updates, as their generation depends on the model's current parameters. The literature shows that by using both clean data and adversarial examples the model can be regularized [47, 58]. In large-scale ML applications such as fraud detection, classifiers require massive amounts of data, thus elevating the cost of this approach. Kurakin et al. [59] propose a solution by doing adversarial training in batches.

Even though most of the literature focuses on neural networks, tree-based models are also a relevant topic to be discussed. These models are not vulnerable to gradient-based attacks, but they are just as vulnerable to adversarial samples. Examples of approaches to address this problem are adversarial training and *robust splitting*. The latter was proposed by Chen et al. [60] and is based on augmenting the standard tree training procedure, increasing the robustness of these models.

2.4 Observational Fairness in Dynamic Environments

In dynamic environments, it may be harder to achieve fairness because the data and the context in which the models are being used can change. However, it is important to take a proactive approach to address these obstacles to ensure ML is being used in a non-discriminatory way. Given that, attention to fairness in dynamic environments has been growing. Some works show the impact of the aforementioned phenomenons on fairness and how to mitigate it. The remainder of this chapter focuses on the available literature studying observational fairness in dynamic environments.

Distribution Shift

If the distribution of the data changes from training to testing time, performance in different groups might be affected, thus influencing fairness. These shifts can take several forms, so different methods are used to identify and mitigate them. Rezai et al. [61] address covariate shift by using a game between an adversary choosing conditional label distributions to fairly minimize the predictive loss on the target distribution and another to maximize the same objective. The authors incorporate fairness as a penalty term in the objective that evaluates fairness in both conditional label distributions. Conversely, Mishler and Dalmasso [62] claim that observational fairness metrics are not accurate when dealing with concept shift; instead, those authors propose the use of metrics that depend on counterfactual outcomes.

Performative Prediction

As performative prediction is a particular case of concept shift, it goes without saying that it can affect fairness. Milli et al. [63] demonstrate that negative externalities of strategic classification can disproportionately harm some groups, thus one needs to weigh strategy-robustness against social welfare and fairness. Another work conducted by Estornel et al. [64] points out the negative impact of adaptive agents in classifiers trained to be fair. Pombal et al. [9] show not only how adaptive fraudsters affect the fairness of predictive models but also study the impact of selective noisy labels on a classifier's fairness. The authors use a group-wise thresholding technique to mitigate its impact. The noisy labels problem is also studied by Fogliato et al. [65], proposing an analysis framework for evaluating how assumptions on the noise across groups affect the predictive bias properties in a criminal justice setting.

Adversarial Attacks

A common problem ML faces is the threat of adversarial attacks. This means one has to build robust models to withstand them, making robustness and adversarial attacks appear hand in hand in the literature. Several methods have been proposed to increase robustness against this kind of attack, for example, adversarial training. In the context of multi-class image classification, it was shown that traditional methods used to obtain robust models led to changes in classification accuracy for different classes, which can lead to fairness problems as studied by Xu et al. [66]. Other authors criticize current definitions of fairness, as they don't account for the robustness of the results with a breakdown per protected group [67, 68]. Conversely, in natural language processing, it has been proven that methods for increasing robustness improve overall fairness (and vice versa) [69].

All in all, literature commonly connects fairness to robustness, as they share common properties. The main difference between them is that while fairness aims to have models invariant to one or more sensitive attributes of the data, robustness has invariance to perturbations as its objective.

2.5 Final Remarks

Fairness is a growing concern for the applications of ML, as models are starting to be used in increasing high-stakes scenarios. This has led to several works in the area of *fair ML*, regarding evaluation metrics, enhancement methods, and trade-off studies. Another relevant topic in the ML literature is approaching dynamic environments, motivating the development of methods to build robust models that are not vulnerable to stressful environmental conditions or adversarial attacks. On the one hand, research often addresses robustness together with either adversarial attacks or fairness. On the other hand, studies addressing robustness and fairness together in stressful environmental conditions are scarce, especially in tabular data settings.

In the following chapter, we outline our proposed solution. Our work focuses on different perturbations on tabular data and their impact on both fairness and performance. Also, the chosen setting includes temporal dynamics, where, for example, the bias conditions change over time. This allows us to test robustness to adversarial attacks in stressful environmental conditions, regarding both fairness and performance.

Chapter 3

A Framework for Evaluating Robustness

FairML studies mostly focus on static environments. The topic of robustness is widely studied in the literature, but it is mostly connected to adversarial attacks and performance. Furthermore, the adversarial attacks are almost exclusively directed at image data. In this thesis, we work with various datasets, however, the task at hand is always binary classification using tabular data. Unlike images, this type of data is heterogeneous, featuring dense numerical features and sparse categorical features [56], providing a different set of challenges when developing models and crafting adversarial attacks.

The objective of this work is to provide a robustness view that takes, not only performance but also fairness into consideration, as well as a a framework to evaluate model robustness using tabular data. We explain the usual dynamics of a real-world environment and how to simulate them. These dynamics feature a range of perturbations that disrupt data. We propose a taxonomy of the most common set of perturbations that occur in the fraud detection setting. We then propose a framework for evaluating robustness that is dependent on the perturbation being studied.

3.1 Overview

In general, supervised learning requires large amounts of labeled data. Data scientists gather available data and use supervised learning to obtain the best possible model according to some objective. For example, in a credit attribution setting, one can use a model to predict whether the customer will pay it back, and the data used to train these models are input-output pairs, where the input is a group of features (like housing status, occupation, etc...) and the output is a binary variable that signals if the credit was repaid. With this in mind, data scientists use their knowledge to build statistical models that, given a new input, aim to correctly predict the outcome. Using appropriate evaluation metrics, a model is finally obtained so that it can be deployed and used in the real world. Here, it is said to be in *production*, and its objective is to be useful to its users, meaning that, given new data, it needs to correctly predict the outcome. This lifecycle of a model is displayed in Figure 3.1.

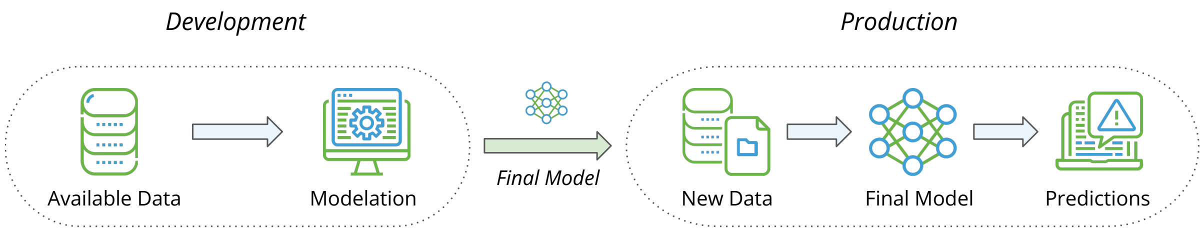


Figure 3.1: Model lifecycle.

Data scientists develop statistical models using data that has a certain distribution, which shapes the input-to-output mapping that is obtained. However, the world is dynamic and complex, so the data that will be used in *production* rarely follows the same distribution as the one used in *development* [70]. Countless factors contribute to this phenomenon, namely data collecting problems, consumer habit change, and malicious intent, among others.

As explained in Chapter 2, when data distribution changes, so does model behavior. Most commonly, the major problem that the entity that deploys the model worries about is performance degradation. However, with growing concerns about responsible AI, these entities should also account for its impact on fairness, especially if the models are used for decisions that have a great impact on individuals' lives. For instance, imagine a bank that operates in a certain country and develops a model that helps in a loan attribution setting, predicting if it will be paid back, using data gathered over the years on that location. The model is deployed and is a success, the bank thrives and begins expanding, opening a new branch in a new country on the other side of the world. If this bank wants to deploy its model in this new country, it needs to be wary that the demographics, data quality and reliability, and even the laws of the location might differ, thus leading to different data distributions. This may lead to unwanted model behavior, namely discrimination. Loan attribution is a high-stakes scenario and considering fairness concerns is vital, making it crucial to evaluate model robustness regarding fairness, not only performance.

With this in mind, we want to test how the models behave in the presence of these changes in distribution and evaluate their robustness, regarding not only performance but also fairness. To mimic this setting, we apply a set of perturbations to the data (thus simulating what happens in the real world) and study their impacts on performance and fairness. The setup for measuring robustness against perturbations can be outlined at a high level as follows (a visual representation is available in Figure 3.2):

1. First, we split the available data in each dataset into three different sets: training, validation, and test.

2. We use the training data to train various models.

3. The validation set is used for evaluating the models and choosing the best one (*final model*) according to a problem-specific metric. In unbalanced datasets, it is also used to tune the decision threshold (that will also be used in test prediction).

4. The test dataset simulates the data in production; this is where we apply our perturbations, resulting in several *perturbed test sets*.

5. We evaluate the robustness of the final model using the predictions of both perturbed and clean test sets.

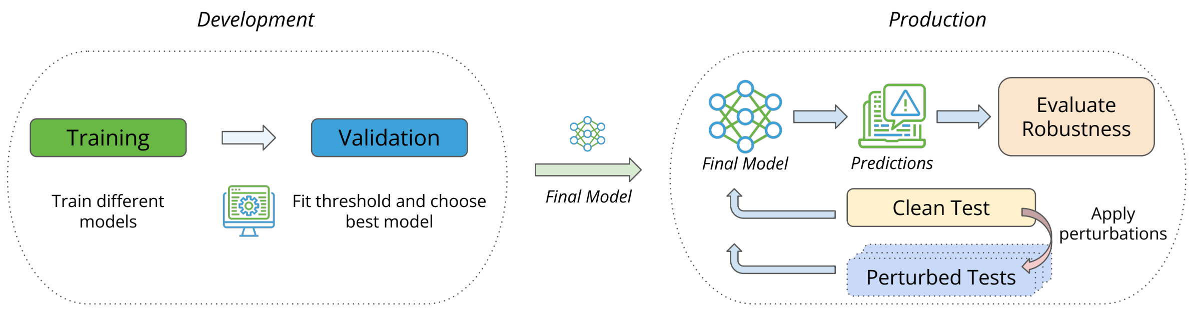


Figure 3.2: Model lifecycle simulation setup.

These first models are developed without any robustness concerns, but it is evaluated nonetheless. Initially, we want to study how models are affected by these perturbations in case they are not taken into account in the model development process. After gathering these results, our focus moves towards developing robust models, so that we can lower the unwanted effects of the perturbations.

When it comes to building robust models, the existing literature leans heavily towards building adversarially robust models. We take advantage of that and focus our process on enhancing adversarial robustness. The setup for building robust models and evaluating robustness is only different from the previously mentioned setup regarding the training process. Finally, we compare the robust models against the corresponding models trained without robustness concerns.

In summary, this thesis starts by investigating the influence of perturbations on both model performance and fairness, comprehensively assessing their robustness. Armed with insights into these impacts, novel models are subsequently developed with the specific goal of boosting their resilience against the previously mentioned perturbations. Finally, the final part of the work offers a comparative analysis, highlighting the efficacy of the newly developed models in their ability to withstand the same set of perturbations when contrasted with their predecessors.

3.2 Perturbation Taxonomy

The real world is a dynamic and complex environment, making it difficult for a model to keep its correctness after deployment. Data distribution continuously changes over time, affecting how models behave. These *shifts* can be caused by intangible reasons, such as consumer habit changes, demographics, seasonal changes, and so forth. Our dynamic environments feature these shifts, but the perturbations that we will include in our taxonomy are tangible ones: causes for drift that we can observe and quantify.

It would be impossible to list every perturbation that a model can suffer in *production*. Given that our setting focuses on tabular data and, more specifically, on financial fraud, we can narrow them down to a more relevant subset. Taking this into consideration, we propose a taxonomy of perturbations for this study consisting of the most common problems that one has to face in this scenario, as shown in Figure 3.3.

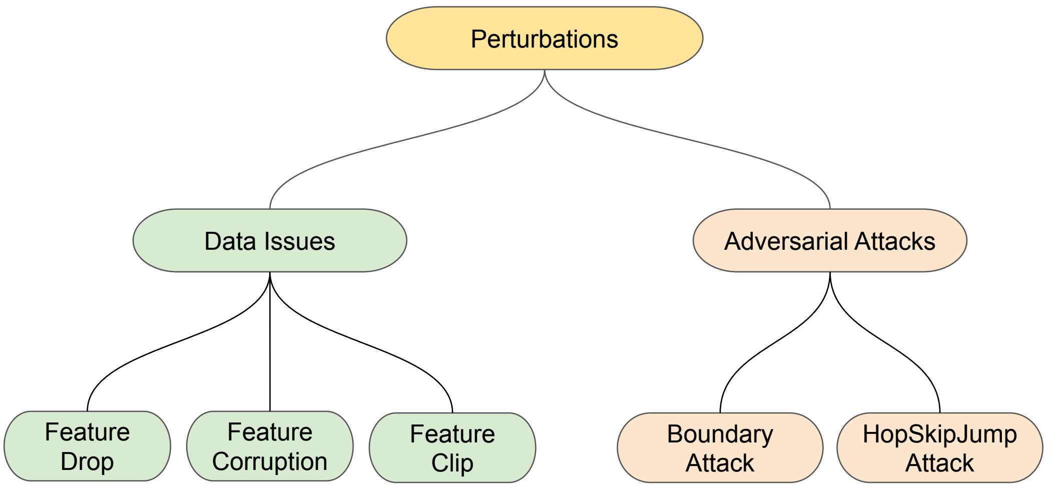


Figure 3.3: Perturbation Taxonomy.

Before diving into studying different problems in detail, we first focus on two major challenges at the heart of our research: adversarial attacks and data issues. These challenges are very important when it comes to detecting financial fraud, and they each come with their unique problems. Adversarial attacks are techniques to fool the best models. On the other hand, data issues cover a range of potential problems, like missing information or strange patterns in the data, which can make our systems less dependable.

3.2.1 Data Issues

Data Issues refer to problems that harm the quality of the data. These issues can be caused by problems in data acquisition or preprocessing done by entities, among others, which may lead to changes in data distribution. Loss of information, introduction of bias, and model instability are some possible consequences of data issues. To better understand model behavior in the presence of data issues, we study the influence of three different perturbations within this category: feature dropping, feature corruption, and feature clipping.

Feature Dropping

Supervised machine learning needs access to training data. In *production*, the models make their predictions on new data. In an ideal scenario, every single instance of data would be filled, with no empty values. Unfortunately, that is unlikely. In most applications, some instances will have missing values, for various reasons, such as data acquisition sensors malfunctioning, incomplete surveys or forms, missing historical data, or changes in reporting standards.

In our work, we study how this kind of perturbation influences model behavior. We perturb each feature individually and evaluate how robust models are in this setting. Furthermore, we suggest dropping values in different ways. We replace values with NaN (not a number), negative imputation values (feature dependant), zeros, and with the mean (for numeric features) and mode (for categorical features). An example of this kind of perturbation is shown in Figure 3.4, where we see that although one feature is perturbed, the remaining ones and the target remain unchanged.

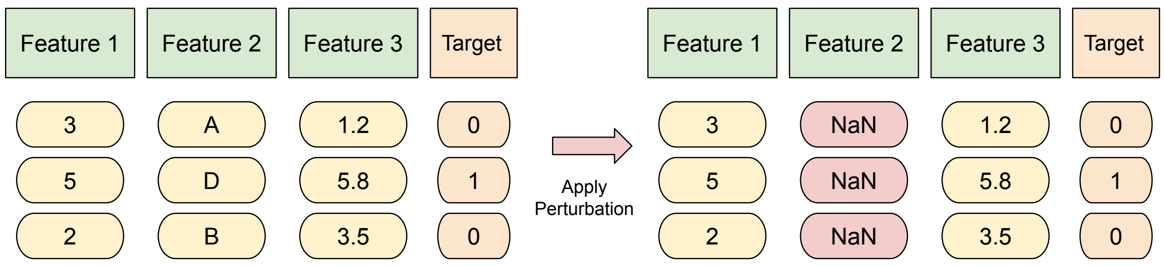


Figure 3.4: Feature Dropping on Feature 2 using NaN as missing values.

Feature Corruption

Another problem that may arise is that, even though a certain value is present, it is meaningless - this is, corrupted. When applied in *production*, models might be using a corrupted feature in their predictions, possibly affecting their performance and fairness. Some reasons that might lead to data corruption are data entry errors, hardware failures, software bugs, and data transfer issues.

To simulate this perturbation, we apply it to each feature individually, shuffling all its values. This way, that feature loses all its relation with the remaining features and with the label - thus becoming meaningless. A depiction of this perturbation is shown in Figure 3.5, where only one feature is perturbed, and the remaining ones are unchanged.

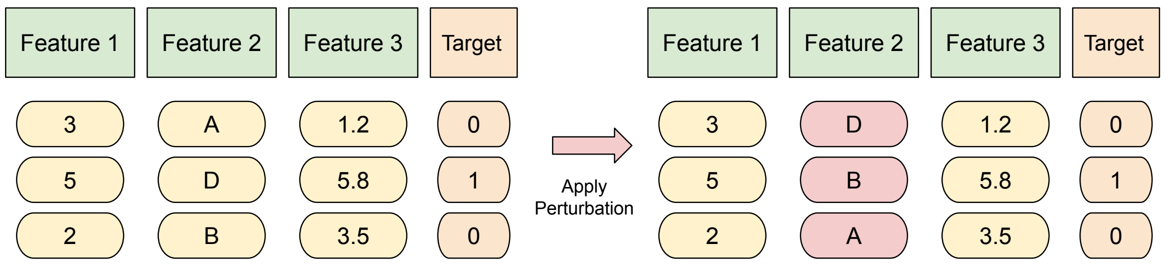


Figure 3.5: Feature Corruption on Feature 2.

Feature Clipping

Instead of completely losing its value, a feature can be simply distorted. When making predictions in production, models need access to their data somehow. Often, data is provided by entities that process it beforehand. These entities may truncate or set lower and upper bounds for some features, altering their distribution. This phenomenon may impact the model's performance. Feature clipping poses a problem as it may lead to information loss and biased predictions, as the distribution no longer reflects the original one.

To study the impact of feature clipping, we target each (numerical) feature individually, clipping values according to upper and lower thresholds. We apply different thresholds to simulate different levels of distortion. Figure 3.6 shows an example.

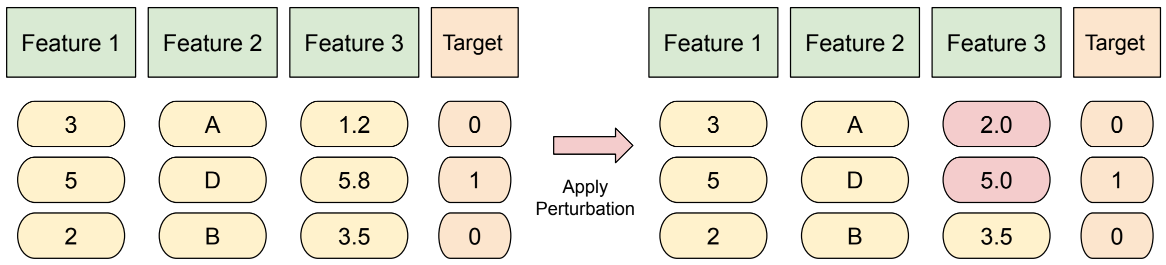


Figure 3.6: Feature Clipping on Feature 3. Here, the lower threshold is considered to be 2.0, and the upper threshold 5.0.

3.2.2 Adversarial Attacks

Adversarial attacks are a category of challenges that pose significant threats to the robustness of ML models. These attacks can be initiated by malicious actors or adversaries aiming to deceive or manipulate the model's predictions, which is undoubtedly a topic of interest in fraud detection. Adversarial attacks can lead to deviations in the model's decision boundaries and undermine its ability to make accurate predictions. Consequences of adversarial attacks include increased vulnerability to fraudulent inputs, a decrease in model accuracy, and potential ethical concerns regarding model fairness and security [71]. In our setting, it makes sense to study *evasion attacks*, where the attacker targets the test set aiming to slightly perturb the input and fool the model. A simplified view of an attack is represented in Figure 3.7, where we see that slight perturbations in the input lead to a change in model prediction.

Furthermore, to better mimic the process of adversarial attack generation in fraud detection, we assume a black-box decision-based setting, where the attacker has no information about the model. In this adversarial attack category, the attacker can only query the model with certain inputs and observe the hard-label binarized output. As the available attacks in the literature are developed for image classification tasks, we adapt two of them to a tabular data setting: Boundary Attack and HopSkipJump Attack.

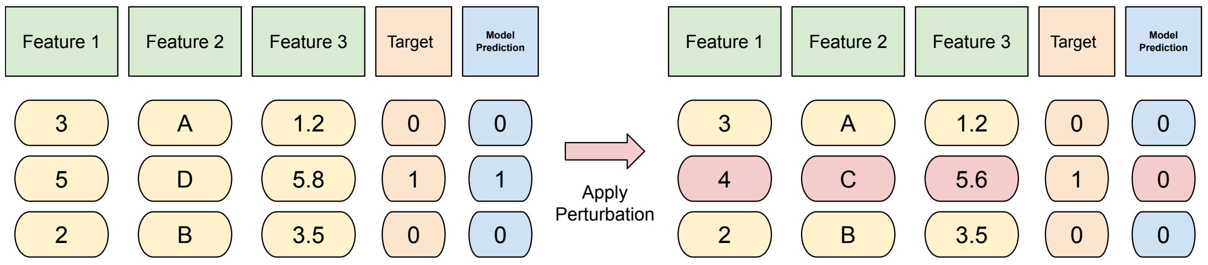


Figure 3.7: Adversarial Attack on the 2nd-row sample, flipping model prediction.

In this work, we want to apply our robustness evaluation framework, which considers fairness. We are that aware tabular data has diverse datatypes, and different feature importances, and some of the fields are often not accessible, which influences how perpetrators forge their attacks, calling for the use of custom norms when measuring the size of perturbations. However, for the sake of simplicity and the possibility of testing multiple datasets (that would require multiple custom norms), we use the l_2 norm, which although does not represent a faithful representation of real-world applications, allows for testing our methods in a high-level approach. The aforementioned chosen attacks have already been proven effective in tabular settings and fraud detection datasets [57], so using them allows for focusing on our robustness framework. Using the l_2 norm means assuming all features are equally important for attack detection and accessible to the adversary.

Boundary Attack

Among the early techniques for black-box decision-based attacks, the Boundary Attack [53] holds a significant place. The primary objective of this attack is to discover the minimal perturbation required to induce a misclassification. Boundary attacks are characterized by their iterative nature: starting with an initial input that is correctly classified, they systematically introduce small modifications while continuously monitoring the model's responses. Instead of the conventional gradient-based optimization approach used in convex problems, the Boundary Attack employs another strategy—specifically, a randomized exploration of the decision boundary. This approach navigates the boundary using a random walk, striving to identify the optimal perturbation for successful misclassification.

HopSkipJump Attack

Another approach to finding the optimal attack aimed to improve the efficiency of the optimization process: the HopSkipJump Attack [54]. The essence of this attack is the same as the Boundary Attack, differing only in the optimization scheme. Instead of using random walks on the boundary, this method changes the optimization problem so that it can use a gradient-based approach, making it more effective.

3.3 Evaluating Robustness

A possible solution for withstanding *distribution shift* is to frequently retrain the model with newer data. Unfortunately, this process is not only expensive but often also ineffective. The alternative is to build robust models. In Chapter 2 we defined model robustness as the resilience against perturbations, meaning a model is robust if it maintains levels of correctness when perturbed. Our approach regarding robustness is based on this definition, thus a model is more robust the more it resists to perturbations.

Robustness is a central topic of this thesis, making it imperative to establish a means of evaluation. However, we argue that robustness should be tailored to the specific problem rather than relying on a universal metric. The ability of a model to withstand one type of perturbation, such as adversarial attacks, differs from its resilience to noisy data. Bearing this in mind, we consistently assess robustness in a manner most suited to the particular problem at hand, thus proposing a framework for evaluating robustness that takes into consideration the perturbation which the model's resilience is being tested against.

3.3.1 Robustness Against Data Issues

To be robust against data issues, a model has to maintain levels of correctness (fairness and performance) when perturbed by them. Given that the injection of data issues into the *clean test set* results in various *perturbed test sets*, a robust model would be one that had similar levels of both fairness and performance on each *perturbed test set* and the *clean test set*. Taking this into account, the evaluation of robustness against these perturbations must be based on the difference in performance and fairness between clean and perturbed sets.

The robustness report of a model against data issues consists then, of differences in the values for the chosen metrics for the problem between *perturbed* and *clean test sets*. For example, if we are evaluating robustness against feature dropping in a setting with 5 features where the chosen evaluation metrics were accuracy and equality of opportunity, for performance and fairness respectively, we would have a difference in accuracy and a difference in equality of opportunity for each one of the 5 features.

3.3.2 Robustness Against Adversarial Attacks

Unlike data issues, adversarial attacks are purposely designed to make models misclassify samples. That said, if we let an adversary perform attacks unconstrained, every attack would flip model predictions. This means that if we attacked the whole test set, we would have an accuracy of 0%. This renders the previous approach for evaluating robustness useless, as values for performance would always be 0, and values for fairness would be unstable (ratios, for example, would be divisions by 0).

A possible solution for this problem would be to add constraints to the attacker so that the success rate wouldn't be 100%. However, this approach would not be problem-agnostic, thus sparking a need for tuning, making it less appealing. Instead, we propose measuring attack effectiveness, sweeping across different constraint values and observing the relations among them. The overall view of this approach is that the harder it is to forge effective attacks, the more robust a model is against adversarial attacks.

A successful adversarial attack is an attack that fools the model, leading it to misclassify the target sample. However, some properties make a successful adversarial attack more effective. An attack that requires fewer queries from the model to be created is more effective than one that needs more queries. Additionally, a perturbation that is smaller, according to some chosen norm, is more effective, as it results in an adversarial sample more similar to the original one [7].

Thus, to evaluate performance robustness, we sweep across different values of the number of queries constraint and register the values obtained for perturbation norms (in successful attacks). Additionally, we fix the number of queries and sweep across various perturbation sizes as constraints, and observe how the success rate evolves. In the latter, an attack is only deemed successful if the perturbation behind it is smaller than the constraint.

Robustness regarding fairness calls for a slightly different approach. If an attack is more effective against one sensitive group than another, it means the model is less robust for that group. This leads to a need to learn how attacks affect each group individually. The metrics used to measure attack effectiveness are the same, but this time we need to take into account groupwise concerns. With this in mind, we present the same results for the perturbation size separately for each group.

Furthermore, we propose two new metrics for evaluating robustness regarding fairness. The first one represents the ratio between perturbation sizes for each group. We will consider the perturbation size measured as an l_p norm for simplicity (any custom norm can also be used).

$$PSR = \frac{\min(l_p^{g1}, l_p^{g2})}{\max(l_p^{g1}, l_p^{g2})}, \tag{3.1}$$

where l_p^{g1} denotes the mean perturbation norm of the attacks against a protected group $g1$ and l_p^{g2} against another group $g2$. The Perturbation Size Ratio (PSR) is the ratio between the sizes, forcing the highest value to be on the denominator, and guaranteeing values in the [0,1] range. The other proposed metric is the ratio between the success rate (SR) for each group: the Sucess Rate Ratio (SRR) is computed as follows:

$$SRR = \frac{\min(SR^{g1}, SR^{g2})}{\max(SR^{g1}, SR^{g2})}, \tag{3.2}$$

where SR^{g1} denotes the mean success ratio of the attacks against a protected group $g1$ and SR^{g2} against another group $g2$. This again forces the highest value to be in the denominator to keep values in the desired [0,1] range. These metrics are useful as they reveal whether a model is more robust for one sensitive group than the other.

Chapter 4

Experimental Setup

4.1 Overview

The proposed robustness evaluation framework considers the effects of perturbations on fairness, which is not given enough attention in the robustness literature. Each experiment explained in this chapter consists of injecting perturbations into the test set and assessing model robustness against them.

A holistic view of our setup could be divided into two steps. The first one simulates model *development*, where we simply train models and choose the best one. The second step, the main focus of our research, is measuring how robust the model is in *production* by applying our framework proposal.

The datasets used in this thesis are always split into, training, validation, and test. The first two are used in the *development* simulation and the latter in the *production* simulation. In the training set, we use the data to train various models. In validation, we evaluate the models and choose the best one. Additionally, if needed, we tune the decision threshold on this dataset as well. Finally, the test set is where we use the robustness framework, applying perturbations and assessing how the model behaves.

4.2 Datasets Description

In this research, we focus on the field of financial fraud detection, where we deal with tabular data for binary classification tasks. We use three distinct datasets, each with its unique characteristics. These datasets encompass a dataset simulating real-world bank account opening fraud, a drift and bias-free dataset we constructed as a reference point against the more realistic one, and a dataset commonly used in the existing literature. Depending on the specific dataset, we employ different fairness and performance metrics. Our goal is to understand how the dynamics and biases inherent in these datasets impact a model's resilience when exposed to perturbations.

Notably, the criteria we use to evaluate model fairness and performance adapt to the specific dataset under consideration. Since fairness and performance can vary significantly based on dataset intricacies, in each dataset, we, not only explore their unique properties but also adjust the evaluation standards to match the nuances of the fraud detection challenges they present. We aim to uncover the complex relationship between dataset attributes, model performance, fairness, and the model's ability to withstand perturbations, unveiling the obstacles financial fraud detection models face in a constantly evolving landscape.

The three datasets in question are presented in the following subsections.

4.2.1 Bank Account Fraud Dataset Suite

The chosen datasets regard the detection of online bank account opening fraud in a large consumer bank - BAF (Bank Account Fraud) [72]. In account opening, fraudsters attempt to impersonate victims via identity theft or create a fictional character to access banking services. If their attempt is successful, they max out the given line of credit or use the account to receive illicit payments. Given that there is no way of tracing the fraudster's true identity, all costs have to be sustained by the bank.

The BAF suite is composed of six datasets that were generated from a real-world online bank account opening fraud detection dataset. Each one of the datasets on the suite was generated by leveraging a state-of-the-art GAN model [73] and each instance of the dataset represents an individual application. The temporal distribution shifts and bias conditions that characterize the datasets allow for testing performance and fairness of ML models meant to operate in dynamic environments. The data comprises eight months of information (February to September).

To study fairness, we need to elaborate on how we should account for its impact. One of the factors that influence the amount in a line of credit is the age of the applicant, and older people are usually associated with higher values. Given that, the portion of fraudulent applications is higher for this group, as it provides fraudsters with bigger profits. On the other hand, the amount of requests is higher for younger people, contrasting with the fraud prevalence. This interesting dynamic makes considering fairness regarding a person's age our choice for this setting, as it may lead the model to learn patterns that lead to discrimination against older people.

BAF Base

The Base Variant of the suite, or BAF Base as we call it, is the dataset that recreates the original real-world online bank account opening fraud detection dataset. Each row corresponds to an application for opening a bank account. This dataset perfectly fits our intentions as it features distribution drift over time, both from natural occurrences (for example changes in client behavior) and fraudsters adapting to better fool the model, as well as the common biases associated with bank account opening that we mentioned before. We present an overview of the dataset in Table 4.1 and some fairness information in Table 4.2 and promptly discuss some of its aspects.

Table 4.1: BAF Base overview.

Dataset	**Number of Records**	**Fraud Rate**	**Duration**	**Number of features**
BAF Base	1M	1.10%	8 months	31

Table 4.2: BAF Base fairness overview.

Dataset	Sensitive Group		Group Size		Prevalence	
	G0	G1	G0	G1	G0	G1
BAF Base	Young (<50y)	Old (≥50y)	81.70%	18.30%	0.83%	2.34%

In this dataset, applications are labeled to be either negative (legitimate) or positive (fraudulent). The fraud rate is only slightly above 1%, meaning the dataset is highly unbalanced. Learning from such an unbalanced dataset poses a challenge [74], for example, if we label every instance as negative, the model would have around 99% accuracy. For the blind eye, it would seem like an excellent model, but it wouldn't predict a single fraudulent instance, that is its objective. With this in mind, this setting calls for a specific evaluation framework that is presented in the final part of this subsection.

Regarding the responsible AI aspects of the datasets, two different kinds of bias are observable: prevalence disparity and group size disparity. The old people group is smaller and has a higher prevalence.

Another important consideration in this dataset is the prevalence variation over time, as represented in Figure 4.1. Globally, there is a prevalence fluctuation over time, and notably the prevalence is at its highest in the last months. When it comes to sensitive groups, prevalence fluctuation differs: the young group has a similar drift to the global fluctuation, with the overall values being slightly lower; on the other hand, for the older group, the variations are ampler and the values are mostly higher than 2%. Anyhow, the months with the highest prevalence are the last two for all three curves.

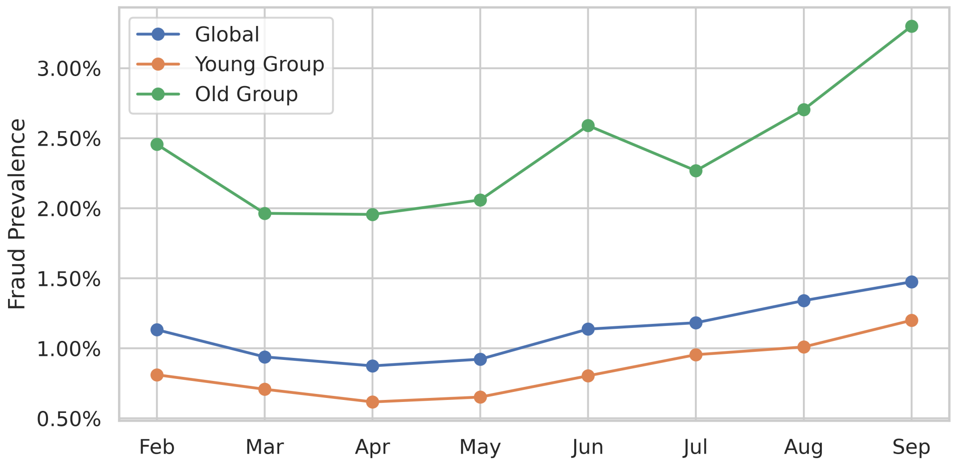
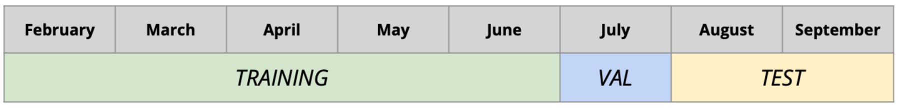


Figure 4.1: BAF Base fraud prevalence over time.

Our setup includes, as we've seen, a train set, a validation set, and a test set, meaning we have to split this dataset into three sets. We use the first 5 months for training, the next one for validation, and finally the last two for testing. The time-based split goes as in Figure 4.2. Given that the prevalence changes over time, each split will have different values. This drift over time is one of the focal characteristics of the environment we wish to study. Particularly, the test set will always have a higher prevalence than both training and validation.

By considering this dataset in our testbench, we can study robustness regarding both fairness and



February	March	April	May	June	July	August	September
TRAINING					*VAL*	*TEST*	

Figure 4.2: Time-based splits for BAF Base.

performance in a stressful environment, where bias and drifts are present.

BAF IID - Generating an IID and unbiased BAF Variant

To better understand the impacts of biased data and distribution shifts, we need a dataset that is free of both, serving as a baseline. The BAF Base dataset has prevalence disparity, group size disparity, and data distribution changes from month to month. In opposition, the baseline needs groups that have the same size and fraud rate, and data distribution doesn't depend on time.

All the variants available in the BAF suite were generated using a state-of-the-art GAN. Instead of following the same process, we simply take advantage of one of the variants and adapt it to better fit our purpose. Knowing that every variant has distribution drift, we use Variant II as it has prevalence disparity but no group size disparity. The steps that are followed to create the independently and identically distributed (i.i.d,) and unbiased baseline are the following:

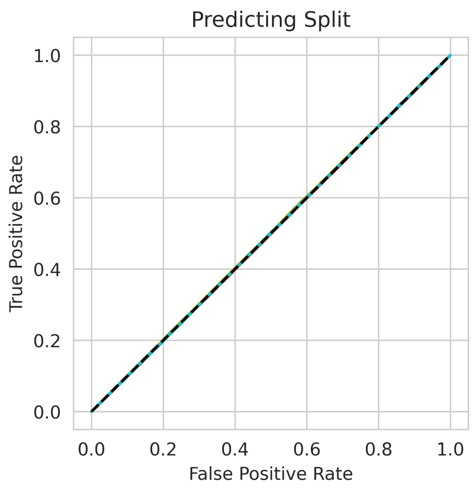
1. Remove the temporal component: drop the *'month'* column. This way, there is no variation over time.

2. Remove the relation of the sensitive attribute with both the remaining features and the label: binarize the *'customer_age'* feature (according to the sensitive group) and then shuffle it.

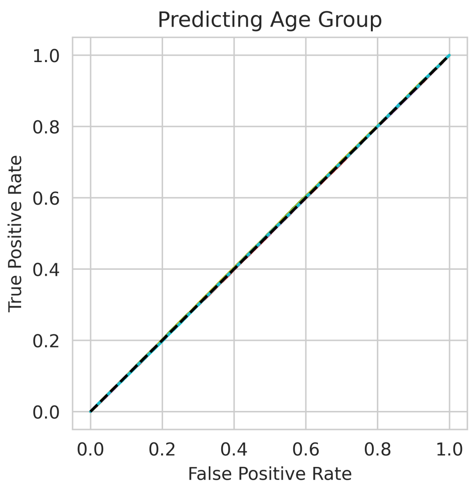
To prove that the created dataset was indeed both i.i.d. and unbiased, we trained 10 models (LightGBM [75]) for two different tasks: (1) predict the split, where in this case, to make it a binary problem, we simply joined the train and validation as the label 0 and used test as the label 1 - if the model can't distinguish between the splits, the splits are i.i.d. (2) predict the binarized sensitive attribute. The training set corresponds to 70% of the dataset and the remaining 30% is used as a test set. If the dataset had the desired characteristics, these models should behave as random classifiers. The results of this experiment are visible in Figure 4.3.

The plots in Figure 4.3 represent the model's ROC curve. When a ROC curve forms a 45-degree diagonal line from the bottom left to the top right, the AUC=0.5, meaning the model is no better than random guessing. In this case, the model is as likely to make correct and wrong predictions. This 45-degree slope represents randomness in the sense there's no meaningful ability to distinguish between the two classes.

As every ROC curve available in the results has a 45-degree slope, we can assume that the models aren't able to extract statistical relations neither between the splits nor the sensitive attribute and the rest of the data. Based on this evidence, we may say the created dataset - BAF IID - has no distribution shift and is unbiased.



(a) IID test



(b) Unbiased test

Figure 4.3: Experiments to confirm dataset characteristics.

With the dataset already generated, we provide an overview in Table 4.3 and some fairness information in Table 4.4.

Table 4.3: BAF IID overview.

Dataset	Number of Records	Fraud Rate	Duration	Number of Features
BAF IID	1M	1.10%	*No Time*	30

Table 4.4: BAF IID fairness overview.

Dataset	Sensitive Group		Group Size		Prevalence	
	G0	G1	G0	G1	G0	G1
BAF Base	Young ($<$50y)	Old ($\geq$50y)	49.40%	50.60%	1.13%	1.08%

Firstly, there is no time component in the dataset, so there is no *'duration'* and, as we've proven, there is no shift amongst train/validation sets and test set. Secondly, we can see that the biases were removed: the groups are approximately the same size and have fairly similar fraud rates. This dataset can then be used as a baseline.

Finally, as there is no time frame, the splits have to be made with random sampling (without replacement), dividing the dataset into three sets: training set (70%), validation set (15%), and test set (15%).

Performance Metrics

Both the datasets that are being used here, BAF Base and BAF IID, are related to bank account opening fraud detection. These datasets are highly unbalanced, only around 1% of the instances are label positives, thus we need to carefully choose how to evaluate model performance in this setting. Choosing the most common metric, accuracy, would not be informative as one could get 99% accuracy by simply predicting every sample as a label negative.

Regarding performance, we base our evaluation process on the receiver operating characteristic (ROC). The ROC curve is a valuable tool in assessing the effectiveness of a binary classification model. By plotting the true positive rate against the false positive rate at various decision thresholds, the ROC curve provides crucial insights into the trade-offs between the two metrics. This guides us in finding the best threshold that leads to the model that best fits the specific goals and requirements of the task at hand. In the bank account opening fraud detection setting, a true positive would be correctly flagging an application as fraudulent, denying the fraudster's attempt to take advantage of the bank, and a false positive would be flagging a legitimate application as fraudulent, denying a regular customer the ability to open a bank account.

Bank account providers are not willing to surpass a certain level of FPR, as each false positive instance may lead to customer unhappiness. This means that our goal is to maximize fraudulent application detection(TPR) without crossing the imposed limit for the FPR, thus avoiding customer attrition. We define the FPR ceiling at 5%.

Fairness Metrics

When enumerating all the observational fairness metrics, we pointed out that its choice is heavily dependent on the task at hand. In our case, the task is bank account opening fraud detection. This is a high-stakes scenario where fairness must be accounted for. A false positive means that an individual was denied a bank account, which might greatly affect his life. On the other hand, a false negative means a fraudster has completed a successful attack, resulting in financial losses for the bank.

Taking this into consideration, we must choose metrics that if enforced in the ML model, ensure that no sensitive group is disproportionately affected. To guarantee that equal proportions of fraud are being caught for each group, we use the *equality of opportunity* metric, hoping to achieve equal FNR (as it means the same as achieving equal TPR). Furthermore, we want to ensure that no group is disproportionately denied access to banking services, so we also use the *predictive equality* metric, aiming to have the same FPR.

However, when putting both these metrics into scrutiny, one can consider that *predictive equality* is a better measure of fairness in this scenario. Enforcing *equality of opportunity* means that the bank is correctly flagging fraudsters equally for both groups, which is not its main ethical concern. The bank account providers want to be fair with their customers, thus the main ethical concern should be to avoid incorrectly denying a customer of any sensitive group the opportunity of opening a bank account, as it would have a big impact on their lives. That said, the main fairness metric in this scenario, and the one we are going to focus on, is *predictive equality*, as it takes into account the false positive ratios.

4.2.2 Adult Dataset - ACS Income

Although his thesis heavily leans towards the setting of financial fraud detection, we considered it would be valuable to test our methods in a widely used dataset in the literature. Concerning tabular data, the fair machine learning research community commonly uses the UCI Adult Dataset [76], based on a 1994 US Census survey. More recently, this dataset was used to build a suite of various datasets, giving researchers more options for fair machine learning works [77]. From this suite, we use ACS Income, where the (binary) task is to predict whether the income of an individual is superior to 50k US$.

Regarding fairness, race was the chosen sensitive attribute, as it is a common source of discrimination in today's society. Also, the ACS Income has no temporal features, so it has no distribution shift over time. Nevertheless, this dataset is useful for measuring robustness to perturbations anyway, as we can apply them to the test set and evaluate their impact, just like we did in the BAF IID dataset. We present an overview of the dataset in Table 4.5 and information regarding fairness aspects in Table 4.6.

Table 4.5: ACS Income overview

Dataset	**Number of Records**	**Prevalence**	**Duration**	**Number of Features**
ACS Income	$\approx$ 1.7M	36.89%	*No Time*	10

Table 4.6: ACS Income fairness overview

Dataset	**Sensitive Attribute**		**Prevalence**		**Group Size**	
	G0	G1	G0	G1	G0	G1
ACS Income	White	Black	39.05%	24.63%	78%	8.9%

Firstly, this dataset is not extremely unbalanced, so evaluation metrics need to be chosen accordingly. There is no duration, as the dataset has no temporal feature. Furthermore, there are fewer features than in the BAF Suite's datasets. When it comes to responsible AI, we will study fairness concerning race, most specifically Black (African American) and White races - even though the dataset has instances of other races. In this regard, there is both group size and prevalence disparity, making it worthy of being used in fairness studies.

As there is no temporal frame in this dataset, the splits are random: training set (60%), validation set (20%), and test set (20%).

Performance Metrics

In the ACS Income dataset, the task is to predict if the income is over 50k US$. Since there is a reasonable number of positive cases (close to 40%), we are not in a situation where simply classifying everything as negative would yield a high accuracy score. Given this, accuracy is a suitable metric for evaluating how well the model distinguishes between different income levels, considering both correct positive and negative predictions, without being overly influenced by the class distribution.

Fairness Metrics

In our research on this dataset's fairness, we emphasize the attribute of race, where we compare the treatment of White and Black (African American) individuals. The choice of race as the fairness attribute stems from the broader societal context where disparities in treatment and opportunities based on race have historically existed. The dataset under scrutiny is tailored for the task of predicting income levels, a task that finds applications across various domains, including credit scoring and risk assessment, policy planning, and more. In this complex landscape, achieving fairness becomes paramount.

In this scenario, both Equality of Opportunity and Predictive Equality emerge as key metrics. These metrics enable assessing the extent to which our models favor or discriminate against certain groups in making decisions like credit allocation. For instance, they help avoid situations where our models inadvertently favor certain racial groups for credit attribution or unjustly reject credit attribution for others. As we delve into our analysis, we recognize the complementary nature of these metrics, and our research endeavors to harness their combined power to comprehensively evaluate and enhance the fairness of machine learning models in income prediction. By employing both Equality of Opportunity and Predictive Equality, we gain a holistic understanding of fairness.

4.3 Model Development Process

4.3.1 Methods and Algorithms

We test two different ML algorithms: LightGBM (LGBM) [75] and feed-forward neural networks trained with the Adam optimizer [78]. The first one is a gradient-boosted tree, a method that has been successful in recent years in the tabular data domain, given its ability to deal with mixed-type data [79]. The other one is a popular ML algorithm and the one most authors use in the robustness literature [7, 58, 80].

The models are trained without any robustness-enhancing techniques. The goal behind this implementation choice is to evaluate how robust models are if no attention is given to robustness in the training process. Furthermore, we want to test if our framework is capable of exposing robustness liabilities any model may have, as that is where its value lies.

Hyperparameter choice is of the utmost importance as it influences both performance and fairness [81]. We use random search, by randomly sampling 50 configurations from a predefined space. The hyperparameter spaces are available in Table A.1 and Table A.2 in Appendix A.

4.3.2 Evaluation and model selection

To choose the best model, we evaluate all the models trained with all the sampled configurations on the validation set. In the case of the Bank Account Fraud datasets, where we need to choose a decision threshold for classification to enforce the 5% FPR ceiling, that threshold is tuned in this set.

The choice of the model here is solely based on performance. This is, the best-performing model on the validation is chosen as the *final model*, which will be evaluated for robustness in the test set. The reasoning behind this implementation rests on the fact that in most real-world applications, the choice of models for deployment is usually based on performance, as stakeholders want to maximize profit.

Chapter 5

Robustness Against Data Issues

When it comes to data issues, the process consists of applying the perturbation to one feature at a time and generating various perturbed datasets. This leaves us with a *clean test set* and a *perturbed test set* for each feature. The framework for evaluating robustness against this kind of perturbation leverages all these datasets available to make its assessment.

5.1 Applying Data Issues

In our perturbation taxonomy, we enumerate three different data issues: feature dropping, feature corruption, and feature clipping. Although implementation is slightly different across them, there are some similarities. Every perturbation is applied to each feature independently, meaning that if we perturb N features, we will generate N perturbed datasets.

Individually, there are also differences between them. Each of the data issue types has different implementations so that we can study the impacts of each variation. To better understand these differences, we describe each class of data issue separately.

5.1.1 Feature Dropping

Feature dropping happens when a feature has missing values. In our setup, we target every sample in the test set, meaning that we will drop that feature from every row of the dataset. Regarding the process of dropping the values, we implement different strategies, as there are different methods for dealing with missing values.

The first approach taken is to replace every value in that column with NaN (not a number). This makes the model treat the value as invalid or empty, affecting how it treats it. In the case of Neural Networks, this method isn't viable, as they reject any non-numeric input. However, LightGBM models accept this kind of value and have a special way of handling them. Values marked as NaN are not used for splits but are then allocated to whichever side reduces loss the most.

Another implementation replaces missing values with the mean of that feature (in case it is a numeric feature) or with the mode (in case it is a categorical feature). The mean and the mode values used are the ones from the data in the training set, to avoid sharing information about the actual feature in the test set. This consists of a common approach in ML to deal with missing values [82]. By using these replacements, we can simulate how a model in production would treat a missing value and then study its impact. This approach works for both Neural Networks and LightGBM.

While the aforementioned missing values treatment was viable for both categorical and numerical features, the two remaining approaches are exclusive for numerical features. One of these approaches is to replace every value with 0. This may happen as, for example, the data that is available instead of marking missing values as NaN, marks them as 0.

The fourth and final method involves employing customized imputation values tailored to each feature. This implies that missing values in each feature are substituted with unique values. For instance, a feature composed solely of positive integers may be assigned an imputation value of -1, while a feature with values within the range [-100, 100] might be imputed with -101. In our implementation, we opt for the first negative value located outside the range defined by the minimum and maximum values observed in the training dataset for the respective feature. However, in cases where the BAF Suite datasheet [72] already specifies imputation values for certain features, we adhere to the values provided in the datasheet.

5.1.2 Feature Corruption

Another type of data issue is feature corruption. This perturbation leads to a feature losing all its value. To lose all its value, a feature needs to lose every relation it has with both the remaining features and the target label.

Our approach to feature corruption involves a straightforward shuffling of the column's values. This disrupts any existing connections between the remaining data and the affected feature, effectively rendering it non-informative during prediction. This strategy serves as a means to assess the model's robustness when confronted with the loss of an individual feature.

5.1.3 Feature Clipping

The third and final data issue in the taxonomy is feature clipping. In this type of data issue, values are truncated or saturated at certain thresholds, causing the data to be distorted. When data is distorted in this manner, distribution changes and may affect model behavior.

In this thesis, values will be clipped by setting an upper and lower threshold on each feature. Naturally, this perturbation can only be applied in numerical features (not boolean). The thresholds are set according to certain percentiles of the training set's feature being perturbed. Our implementation tests two different pairs of values: using the 10th and 90th percentiles, as lower and upper thresholds, respectively, or the 5th and 95th percentiles.

5.2 Evaluating Robustness

The evaluation of robustness against data issues assumes that we have a *clean test set* and some *perturbed test sets*, each corresponding to a copy of the original test set but where one feature has a data issue. The model will compute predictions in each of these datasets, and those will be subject to an evaluation pipeline that leads to the final robustness report.

Each problem has evaluation metrics, for fairness and performance, that need to be measured. For example, we saw that the bank account opening fraud setting considers TPR@FPR=5% and predictive equality, while the prediction of the annual income measures the accuracy and both equality of opportunity and predictive equality. We make predictions on all the available test sets and, to test the robustness of our method, we bootstrap the test 50 times (where each instance is the size of the original and sampled with replacement) and for every instance, compute the desired metrics. This allows us to get a mean value and a confidence interval (CI) of 95%.

The evaluation metrics for robustness against data issues are the differences between *perturbed* and *clean test sets'* performance and fairness. Then, we also want to get the mean and CI for the differences, using the 50 bootstraps. The values for the differences (mean, lower, and upper bounds of the CI) are computed by subtracting the mean value obtained in the *clean set* from the corresponding value in the *perturbed set*. The differences of any given metric x, the mean, and the confidence interval's lower and upper bounds, are computed as

$$\Delta\bar{x} = \bar{x}_{pert} - \bar{x}_{clean}, \tag{5.1}$$

$$\Delta x^{low} = x^{low}_{pert} - \bar{x}_{clean}, \tag{5.2}$$

$$\Delta x^{up} = x^{up}_{pert} - \bar{x}_{clean}. \tag{5.3}$$

These values allow for testing the variability of our method and evaluating the robustness of the model against each data issue applied. If the difference is negative, the lower it is, the less robust the model is, as it loses performance or fairness from *development* to *production*. On the other hand, if the difference is non-negative, the model maintains (or increases) levels of correctness, meaning it is robust.

In conclusion, the results of model robustness against data issues are presented as differences (between *perturbed* and *clean test sets*), with the mean value and 95% confidence interval obtained in the bootstrapping process. Figure 5.1 shows the overall process of evaluating robustness against data issues. To better visualize the results, we present the results in tables, with the following information: number of *perturbed test sets* created (how many features were perturbed), number of those *perturbed test sets* where the entire confidence interval of the metric (each table will represent one metric) is outside the confidence interval of that same metric's confidence interval in the *clean test set*, under the lower bound, and also the mean and lowest values of the metric's mean value of all *perturbed test sets.*

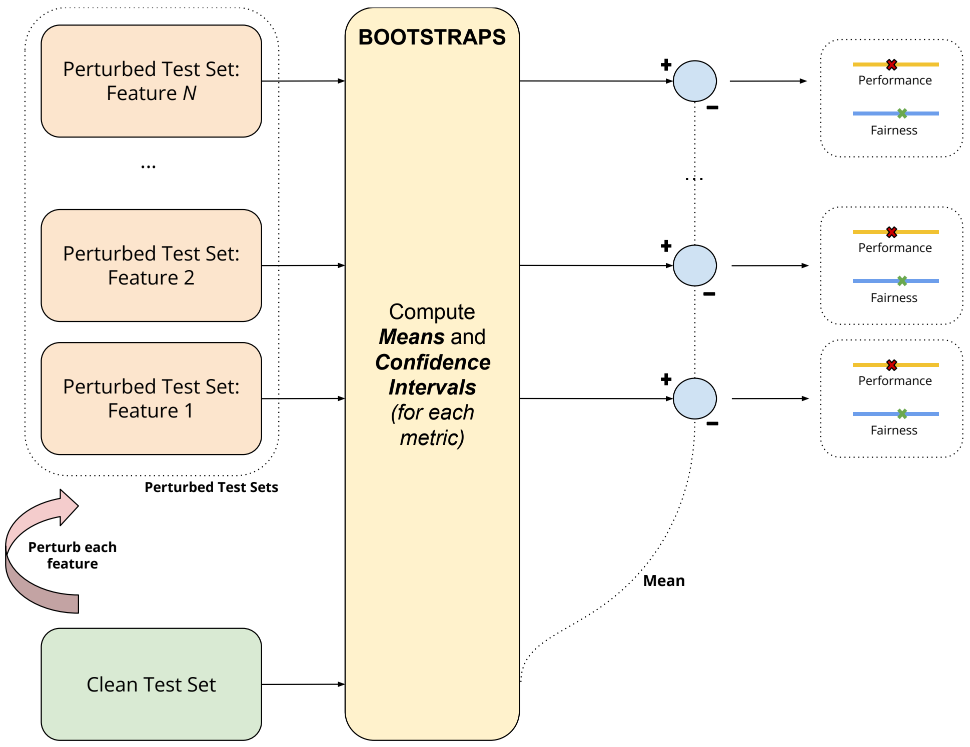

Perturbed Test Set: Feature N
...
Perturbed Test Set: Feature 2
Perturbed Test Set: Feature 1
Perturbed Test Sets
Perturb each feature
Clean Test Set
BOOTSTRAPS
Compute Means and Confidence Intervals (for each metric)
Mean
Performance
Fairness

Figure 5.1: Framework for evaluating robustness against data issues. Each feature in the *clean test set* is perturbed, generating a different *perturbed test set*. The model uses all the available datasets for prediction, and, with various bootstraps, the mean and confidence intervals for each chosen evaluation metric are computed. The deltas' values are calculated by subtracting the mean of the corresponding metric in the *clean test set* from each *perturbed test set*. The robustness results are presented in the form of a 95% confidence interval and the mean value for the delta of each metric.

5.3 Results

This section reports the results obtained in the test set, which mimics the *production* phase of an ML model. The results are divided by dataset and, for each of them, we provide a table with *a clean test set* performance and fairness and a table for the differences of each evaluation metric (fairness and performance) obtained using the *perturbed test sets*. For each dataset, we display the results for one implementation of each proposed data issue category. Regarding feature dropping, we chose to display the results for the mean and mode replacement, as it can be implemented on both neural networks and LightGBM, as well as it can perturb every feature, numerical and categorical. In feature clipping, we chose the more strict thresholds, the 10th and 90th percentiles. As for the remaining implementations, the results are shown in Appendix B.

5.3.1 BAF IID

This dataset, the BAF IID, is unbiased and has no drift among splits, meaning train, validation, and *clean test set* are i.i.d. Table 5.1 shows the results obtained in the *clean test set*. Notably, while performance is higher in the LightGBM than in the Neural Network, fairness is higher in the latter. Fairness values are high for both models, with their confidence intervals standing completely above 90%, as expected for models trained in a training set where the bias was removed.

Table 5.1: Results on the *clean test set* - BAF IID.

Dataset	Model	TPR		FPR Ratio	
		Mean	CI	Mean	CI
BAF IID	LGBM	**52.0%**	[49.5%, 54.3]	**94.9%**	[90.6%, 98.6%]
	NN	**47.1%**	[45.0%, 49.3%]	**98.3%**	[96.1%, 99.7%]

Regarding the robustness study, the results for performance are available in Table 5.2.

Table 5.2: Data Issues impact on BAF IID' performance. The table shows how many *perturbed test sets* are created and how many have an average TPR under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average performance difference and lowest difference.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	29		29		24	
Perturbed Sets w/ CI < clean CI	13	9	3	4	0	1
Average Delta	-4.07 pp	-3.27 pp	-1.98 pp	-1.99 pp	-0.36 pp	-0.59 pp
Lowest Delta	-23.61 pp	-14.92 pp	-10.2 pp	-9.85 pp	-3.15 pp	-4.66 pp

Feature dropping emerged as the perturbation to which both models exhibited the least resilience, leading to performance dropping below the clean confidence interval in 13 different *perturbed test sets* for the LGBM and 9 for the NN. Additionally, LGBM records the lower average (-4.07 pp) and minimum delta values (-23.61 pp). As for feature corruption, 3 and 4 sets drop below the clean confidence interval, for LGBM and NN respectively. Furthermore, the average and lowest delta values exhibited are remarkably similar across both models. Feature clipping stands out as the perturbation with the least impact, making it the perturbation which the model is the most robust against. However, neural networks exhibit slightly less resilience in this scenario, with 1 instance falling outside the clean confidence interval, in contrast to the single set in the case of LGBM. Additionally, the values for both the average delta and lowest delta (-0.59 pp and -4.66 pp) are comparatively lower.

Table 5.3: Data Issues impact on BAF IID' fairness. The table shows how many *perturbed test sets* are created and how many have an average FPR Ratio under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average fairness difference and lowest difference.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	29		29		24	
Perturbed Sets w/ CI < clean CI	0	0	0	0	0	0
Average Delta	0.31 pp	-0.21 pp	0.21 pp	-0.14 pp	0.01 pp	-0.01 pp
Lowest Delta	-2.07 pp	-1.06 pp	-1.75 pp	-1.85 pp	-0.29 pp	-0.2 pp

In the context of fairness (results in Table 5.3), both models have demonstrated remarkable resilience against all perturbations. Notably, no instance deviated from the clean confidence interval in any scenario, and the values for the average delta remain consistently close to zero. It is essential to bear in mind that these models were initially trained on an unbiased dataset and exhibited fairness values exceeding 94% in the *clean test set*. Furthermore, even the lowest delta observed across all experiments, standing at -2.07 percentage points, is not sufficient to push fairness below the 90% threshold.

5.3.2 BAF Base

Unlike the previous dataset, the BAF Base is not unbiased and there is distribution drift among splits. We've seen that the test set has higher fraud prevalence than the remaining splits, for example. The results are displayed in Table 5.4.

Table 5.4: Results on the *clean test set* - BAF Base.

Dataset	Model	TPR		FPR Ratio	
		Mean	CI	Mean	CI
BAF BASE	LGBM	**56.50%**	[54.80%, 58.19%]	**31.3%**	[30.3%, 32.4%]
	NN	**49.3**%	[48.19%,50.4%]	**34.1**%	[33.2%, 35.3%]

The LightGBM outperforms the neural network once again, and overall performance is higher in this dataset than in the unbiased and i.i.d. baseline for both models. On the other hand, fairness is lower than in the prior dataset, as expected. Modelwise, fairness values are similar, with the values for the FPR ratio dropping around 67% and 65% from the ones obtained in BAF IID, respectively. The results for performance robustness against data issues are displayed in Table 5.5.

Table 5.5: Data Issues impact on BAF Base' performance. The table shows how many *perturbed test sets* are created and how many have an average TPR under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average performance difference and lowest difference.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	29		29		24	
Perturbed Sets w/ CI $<$ clean CI	12	9	5	7	0	2
Average Delta	-4.33 pp	-1.69 pp	-1.89 pp	-1.55 pp	-0.56 pp	-0.46 pp
Lowest Delta	-23.71 pp	-20.74 pp	-10.48 pp	-10.12 pp	-2.86 pp	-5.6 pp

The perturbation to which the models are the least robust is feature dropping, resulting in 12 and 9 TPR values falling below the *clean test set* confidence interval for LGBM and NN, respectively. For this specific perturbation, the models also exhibit their lowest values for both average and lowest deltas, registering at -4.33 percentage points and -23.71 percentage points, respectively, with both of these records attributed to LGBM. Feature corruption emerges as the subsequent perturbation with the most effectiveness, causing a decline in performance that falls below the confidence interval for 5 instances in the case of LGBM and 7 instances for NN. Additionally, the LGBM model records the lowest values for both the average delta and lowest delta, which are observed at -1.89 percentage points and -10.48 percentage points, respectively. As for feature clipping, it reaffirms its status as the least impactful perturbation, affecting only 2 cases for NN and none for LGBM, with average delta values in proximity to zero percentage points.

Turning our attention to fairness, it's important to note that these models were not trained under unbiased conditions. The results are visible in Table 5.6.

Consequently, as anticipated, the models' resilience to fairness issues is notably diminished, particularly evident when we consider feature dropping. Within this specific perturbation, a total of 10 instances for LGBM and 4 instances for NN deviated below the *clean test set* confidence interval, and a lowest

Table 5.6: Data Issues impact on BAF Base' fairness. The table shows how many *perturbed test sets* are created and how many have an average FPR Ratio under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average fairness difference and lowest difference.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	29		29		24	
Perturbed Sets w/ CI $<$ clean CI	10	4	1	2	1	0
Average Delta	-1.07 pp	-0.23 pp	0.35 pp	0.56 pp	-0.03 pp	0.07 pp
Lowest Delta	-10.74 pp	-13.25 pp	-4.75 pp	-3.35 pp	-2.29 pp	-1.44 pp

delta of -10.74 percentage points was recorded for LGBM. These models exhibit greater resilience when confronted with the other two types of data issues. Specifically, LGBM's fairness metrics fall below the clean confidence interval in 1 instance for feature corruption and feature clipping. In the case of NN, these metrics drop below the clean confidence interval in 2 instances for feature corruption and none for feature clipping.

5.3.3 ACS Income

The third dataset, ACS Income, is a dataset with no time component, so there is also no drift from split to split. Regarding fairness concerns, this dataset is biased. Table 5.7 shows the results obtained on the *clean test set*.

Table 5.7: Results on the *clean test set* - ACS Income.

Dataset	Model	TPR		FPR Ratio		TPR Ratio	
		Mean	CI	Mean	CI	Mean	CI
ACS Income	LGBM	**82.1%**	[82.0%, 82.3%]	**65.60%**	[63.1%, 69.1%]	**84.7%**	[83.6%, 86.3%]
	NN	**80.5%**	[80.4%, 80.7]	**62.6%**	[61.1%, 65.2%]	**80.4%**,	[78.9%, 81.8%]

The neural network's performance and fairness are lower than the LightGBM. Regarding fairness metrics, we can observe that the FPR Ratio is lower than the TPR Ratio for both models. When taking performance into account, this dataset is evaluated using accuracy, and it's noticeable that variability is low, as the confidence interval has a range of 0.3% for both models. Table 5.8 displays the results for robustness regarding performance.

Table 5.8: Data Issues impact on ACS Income' performance. The table shows how many *perturbed test sets* are created and how many have an average accuracy under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average performance difference and lowest difference.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	9		9		1	
Perturbed Sets w/ CI $<$ clean CI	8	5	8	6	0	0
Average Delta	-1.55 pp	-0.78 pp	-2.66 pp	-1.91 pp	-0.1 pp	-0.03 pp
Lowest Delta	-6.13 pp	-2.84 pp	-8.4 pp	-8.12 pp	-0.1 pp	-0.03 pp

In this scenario, the model exhibits the least resilience against feature corruption, resulting in 8 instances for LGBM and 5 instances for NN where performance falls below the *clean test set*'s confidence interval. Furthermore, for LGBM, the values for both average and lowest deltas are the lowest, standing at -2.66 percentage points and -8.4 percentage points, respectively. Following closely behind is feature dropping, resulting in 8 instances for LGBM and 6 instances for NN falling below the confidence interval. Once more, LGBM exhibits the lowest values, recording an average delta of -1.55 percentage points and a lowest delta of -6.13 percentage points. Regarding feature clipping, as this dataset only has 1 numerical feature, only one *perturbed test set* is created. This perturbation does not significantly impact the model, as the delta value remains in close proximity to zero, and performance remains within the clean confidence interval.

The results for fairness robustness are evaluated based on two metrics, FPR Ratio and TPR Ratio. These results are available in Table 5.9 and in Table 5.10.

Table 5.9: Data Issues impact on ACS Income's fairness. The table shows how many *perturbed test sets* are created and how many have an FPR Ratio under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average fairness difference and difference delta.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	9		9		1	
Perturbed Sets w/ CI < clean CI	1	1	0	0	0	0
Average Delta	1.65 pp	1.65 pp	3.64 pp	2.71 pp	0.53 pp	-0.06 pp
Lowest Delta	-6.45 pp	-4.14 pp	-1.41 pp	-1.31 pp	0.53 pp	-0.06 pp

In Table 5.9, we observe the results regarding the deltas for the FPR Ratio. In terms of fairness, both models demonstrate resilience against feature corruption and feature clipping, with no features falling below the confidence interval of the *clean test set*. However, for the feature dropping perturbation, 1 instance falls outside the confidence interval for both LGBM and NN. This suggests that the models are less robust in this particular scenario. The average delta remains consistent across both models, with the lowest delta recorded for LGBM, measuring at -6.45 percentage points.

Table 5.10: Data Issues impact on ACS Income's fairness. The table shows how many *perturbed test sets* are created and how many have a TPR Ratio under the *clean test set* confidence interval (CI). Additionally, it also presents the values for the average fairness difference and difference delta.

	Feature Drop - Mean/Mode		Feature Corruption		Feature Clip - 10/90	
	LGBM	NN	LGBM	NN	LGBM	NN
Perturbed Sets Created	9		9		1	
Perturbed Sets w/ CI < clean CI	2	0	2	1	0	0
Average Delta	0.02 pp	0.95 pp	-0.46 pp	0.09 pp	-0.01 pp	-0.01 pp
Lowest Delta	-3.05 pp	-0.87 pp	-2.84 pp	-2.86 pp	-0.01 pp	-0.01 pp

Keeping our focus on fairness, Table 5.10 illustrates the results for the deltas of the TPR Ratio. The models showcase their highest level of robustness in the case of feature clipping, with the sole perturbed set maintaining a TPR ratio within the confidence interval of the *clean test set*. Furthermore, the delta value associated with this perturbation is almost negligible, standing at a mere -0.01 percentage points. In the presence of feature corruption, the TPR value falls outside the confidence interval of the *clean test set* for 2 instances for the LGBM and 1 for the NN. Notably, both models record similar values for both average and lowest deltas. In the context of feature dropping, the NN model demonstrates robustness, as no feature falls below the specified values. Conversely, LGBM exhibits a decrease in fairness below the clean confidence interval in 2 instances, even though the average delta remains at a marginal 0.02 percentage points.

5.4 Conclusion

We studied the impact of various perturbations on distinct models using our framework. Notably, different perturbations exhibited varying effects, depending on the model and the dataset. In our analysis, we observed that models showed the highest robustness in the presence of feature clipping, contrasting with a vulnerability to feature dropping. The framework we proposed facilitates a comparison of the impacts of different perturbations, thereby enabling an evaluation of model robustness. For example, our findings revealed that the LGBM displayed comparatively lower resilience against feature dropping compared to NN.

Chapter 6

Adversarial Robustness

Adversarial attacks are the most common perturbations studied in the literature. They represent carefully engineered changes to the input so that they fool the target model. Usually, the main focus of adversarial robustness works is performance. We differentiate ourselves by considering the fairness aspects of robustness.

6.1 Applying Adversarial Attacks

To better fit our setting, which is financial fraud detection, we implement two black-box hard-label attacks: Boundary Attack [53] and HopSkipJump Attack [54]. These attacks assume the adversary can query the model by modifying its inputs and then have access to the hard-label output that is produced. The querying process is used by the method to optimize the attacks and make the perturbations as small as possible.

An adversarial attack, in fraud detection, corresponds to a fraudster trying to fool a model into not recognizing a fraud instance. In other words, it corresponds to a positive labeled instance that is predicted as a negative one by the model, a false negative. With this in mind, the instances that will be targeted to create adversarial attacks will be true positives, aiming to turn them into false negatives.

In our implementation, we target 1,000 true positives (or every true positive, in case there are fewer than 1,000) in the test set, to get reliable results about the effectiveness of our methods. We apply our attack algorithms to all of these instances. We use the default parameters defined by the authors and set a max query threshold of 10,000 queries.

The attacks aim to minimize the distance between the original sample and the adversarial sample generated (finding the smallest perturbation). Although not useful in real-world applications of tabular data, we use the l_2 distance proposed by the authors, assuming attackers have access to all features and every feature is as important for the detection of an adversarial attack. The usefulness of using this norm is that we can see that the methods converge, are able to fool models and behave differently for different sensitive groups.

Both the Boundary Attack and the HopSkipJump Attack were developed for image data, thus they only operate with continuous data. On the other hand, our data is tabular and heterogeneous, with categorical, discrete, and boolean features as well as continuous ones. This means that to apply a perturbation, we need to convert the data to a continuous format. Our approach was to develop an encoder, one-hot encoding categorical features, and scaling numerical features to a range of [0, 1]. After adding a perturbation, the values are clipped to that same range.

While the attacks are designed for continuous data, the black-box models we are attacking use data with mixed data types. This means that, before querying the model, we need to convert the perturbed instance back to its original format. For categorical features, we choose the *dummy feature* with the highest value as the category, and for numerical features, we scale the values back to the original range. In the particular case of discrete and boolean features, the values are rounded to the nearest integer.

Regarding the implementation of the attack methods, we base our work on the available implementations for image data on the Adversarial Robustness Toolbox [83] and adapt them for tabular data by making the aforementioned adjustments. The reasoning behind some of the adaptations of these attacks to tabular data is based on the work by Cartella et al. [57] described in Chapter 2.

6.1.1 Evaluating Robustness

For evaluating robustness, we need information about each iteration of the attacking methods regarding the number of queries made and the current best l_2 norm. This way, we can sweep over them and compute the metrics we mentioned to describe the effectiveness of the adversarial attacks: success rate, l_2 norm (perturbation size), and number of queries.

Regarding performance, we evaluate robustness using the following information: we fix the success rate at 100% (remove the norm constraint - every attack that flips the label is deemed as successful) and see how changing the number of queries influences the perturbation size, measured by a norm. The bigger the norm (the more perceptible the perturbation) for the same number of queries, the more robust the model is. The other approach fixes the number of queries and, by sweeping over the l_2 norm constraint values, evaluates the success rate (here, an attack is only successful if the perturbation norm is smaller than the constraint). The higher the success rate for the same norm constraint, the less robust the model is.

For robustness in regards to fairness, we show l_2 norm variation for each sensitive group, to expose if the model is behaving differently for different groups. An important statement to be noted is that we are only targeting true positives with our attacks, meaning the FPR Ratio (predictive equality) - which is the main metric for measuring fairness in the BAF datasets - won't be affected. With this in mind, we evaluate fairness by using the metrics proposed in Chapter 3: Perturbation Size Ratio (PSR - 3.1), Success Rate Ratio (SRR - 3.2).

6.2 Results

The outcomes in this section pertain to the test set, reflecting the deployment phase of machine learning models. We organize these results by dataset, and for each dataset, we offer visual representations encompassing perturbation size and success rate. These graphical representations facilitate a comparative analysis of the two applied adversarial attack methods, shedding light on their effectiveness. Furthermore, these visuals enable a comprehensive exploration of fairness robustness, highlighting the ratios between perturbation norm and success rate among the two groups being studied in each case.

6.2.1 BAF IID

In this dataset, the test set to which we apply the adversarial attacks has the same distributions as both the train and validation sets. Furthermore, the dataset has no bias regarding the sensitive group (age). The plots in Figure 6.1 allow us to study the adversarial robustness of models developed in these conditions.

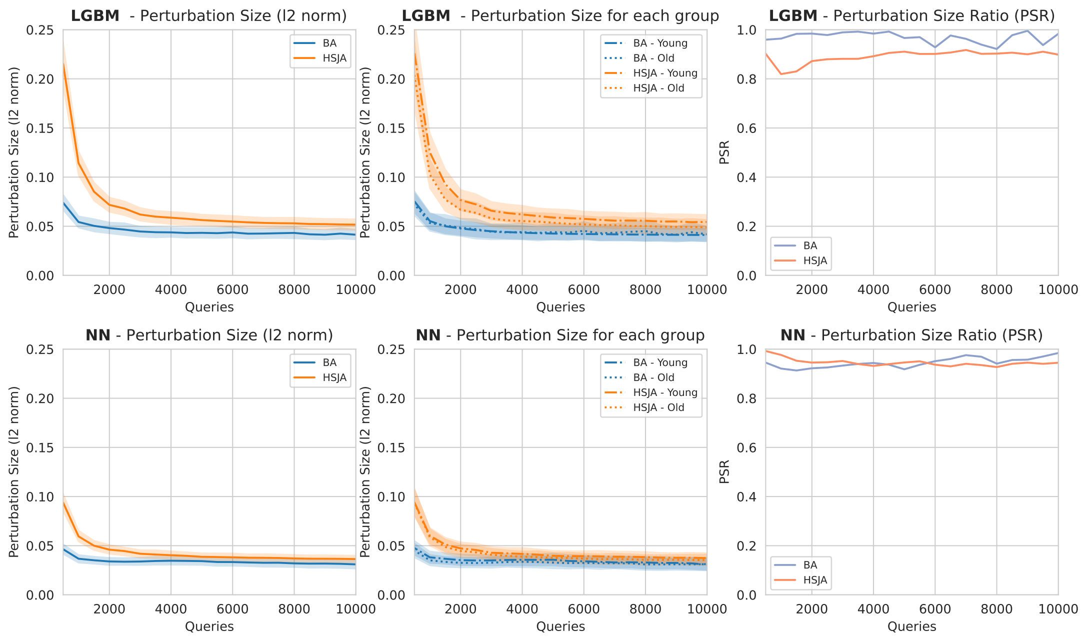


Figure 6.1: Applying adversarial attacks on BAF IID. Evaluating robustness regarding performance and fairness by measuring perturbation size (l_2 norm). The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column plots represent a comparison between the Boundary Attack (BA) and the HopSkipJump Attack (HSJA)'s perturbation norms, the second column separates the previous plots per sensitive group and the third plot shows a comparison of the Perturbation Size Ratio for the two attacks.

The Boundary Attack surpasses the HopSkipJump Attack in performance for both models. Notably, the best l_2 norms achieved are similar and close to l_2=0.05. When we segment the $l_2(queries)$ plot by groups, a consistent observation emerges: the Boundary Attack results in nearly identical l_2 values for both models. Conversely, this phenomenon is only observed with the HopSkipJump Attack in the case of neural networks. When using this attack against the LightGBM, there is a noticeable, albeit slight, difference between groups. The attack proves to be somewhat more effective on the older group, implying that the model exhibits a greater degree of robustness for younger individuals. The PSR thoroughly illustrates this concept. In the case of the neural nets, both ratios consistently hover close to 1.0, which symbolizes a state of complete fairness. This holds true for the Boundary Attack against the LGBM as well. The sole exception is observed when using the HopSkipJump Attack against the LightGBM, which exhibits a slight decline in PSR (Perturbation Size Ratio) when the model is queried fewer than 2,000 times. Each line features some fluctuations, that are primarily due to the small l_2 norm values, where even minor variations lead to noticeable changes in the ratios.

In Figure 6.2, we plot the success rate of the attacks if we add a norm constraint (for instance, we only consider an attack successful if it flips the model decision and as a perturbation norm smaller the the constraint). As we can observe, both models exhibit comparable behavior, with the primary distinction being that the NN converges to a near-perfect success rate slightly more swiftly. For each perturbation norm tested, the Boundary Attack consistently achieves a higher success rate compared to the HopSkipJump Attack on both models.

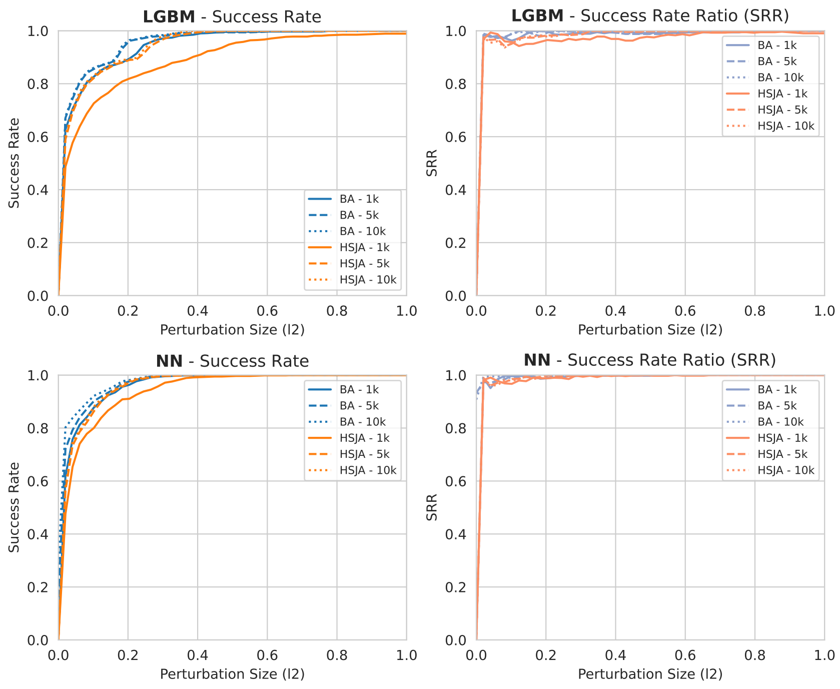


Figure 6.2: Applying adversarial attacks on BAF IID. Evaluating robustness regarding performance and fairness by measuring the success rate. The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column of plots displays the success rate for each perturbation size constraint, fixing a number of max queries constraint (1,000; 5,000; 10,000), for the Boundary Attack(BA) and the HopSkipJump(HSJA) Attack. The second column of plots shows a comparison of the Success Rate Ratio for the two attacks, with the same query constraints.

In terms of the Success Rate Ratio (SRR), once the l_2 constraint allows for successful attacks, the ratio approaches 1.0, signifying an equal success rate for both groups. This observation emphasizes the model's consistent robustness across both groups, in line with expectations for a model developed on an unbiased training dataset.

6.2.2 BAF Base

Unlike the previous dataset, BAF Base the distribution is not the same in each split. Furthermore, there is bias associated with the sensitive feature (age). The results obtained for the l_2 norm are available in Figure 6.3.

Notably, despite sharing the same feature space as the BAF IID dataset, the perturbations here have higher l_2 norms. In this dataset, both attacks reach their optimal solution sizes at approximately l_2=0.65, whereas in the previous dataset, this convergence occurred at around l_2=0.05. Furthermore, the Boundary Attack demonstrates slightly greater effectiveness against the LGBM model, whereas with the neural nets, starting from 2,000 queries onward, the attacks are equally effective.

When we create separate $l_2(queries)$ plots for each sensitive group, the values achieved for the old group are marginally lower, indicating that the models exhibit slightly less robustness for this specific group. Finally, despite these notable differences in the l_2 norms between groups, the Perturbation Size Ratios (PSR) remain relatively stable and closer to 1.0 compared to the BAF IID experiment, primarily due to the substantially higher norm values.

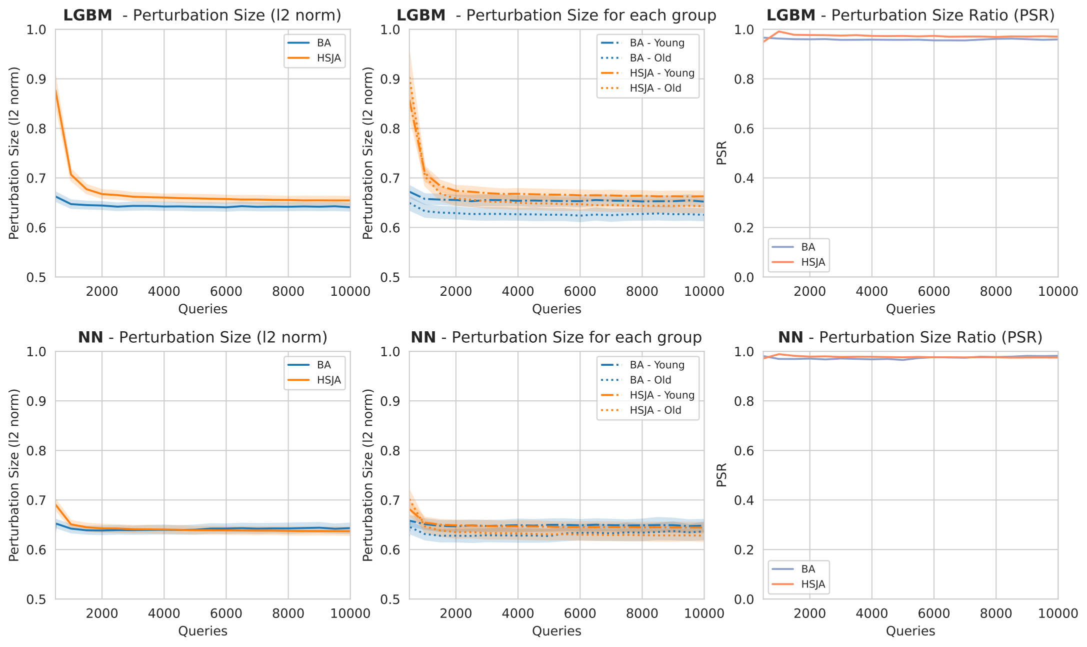


Figure 6.3: Applying adversarial attacks on BAF Base. Evaluating robustness regarding performance and fairness by measuring perturbation size (l_2 norm). The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column plots represent a comparison between the Boundary Attack (BA) and the HopSkipJump Attack (HSJA)'s perturbation norms, the second column separates the previous plots per sensitive group and the third plot shows a comparison ofthe Perturbation Size Ratio for the two attacks.

In Figure 6.4, the success rate of adversarial attacks against models trained in this dataset is shown, as well as the Success Rate Ratio (SRR). In the case of LGBM, the Boundary Attack achieves a higher success rate than the HopSkipJump Attack, and it converges to a complete success rate slightly faster. For the neural networks, the Boundary Attack results in higher success rates for perturbation norms below l_2=0.75, and beyond this threshold, both methods demonstrate comparable levels of success.

Another intriguing finding is the abrupt increase in the success rate at approximately l_2=0.75. This phenomenon may be linked to certain categorical features that have been one-hot encoded in the dataset, suggesting that the spike might be associated with a perturbation significant enough to cause a change in the category of a one-hot encoded feature.

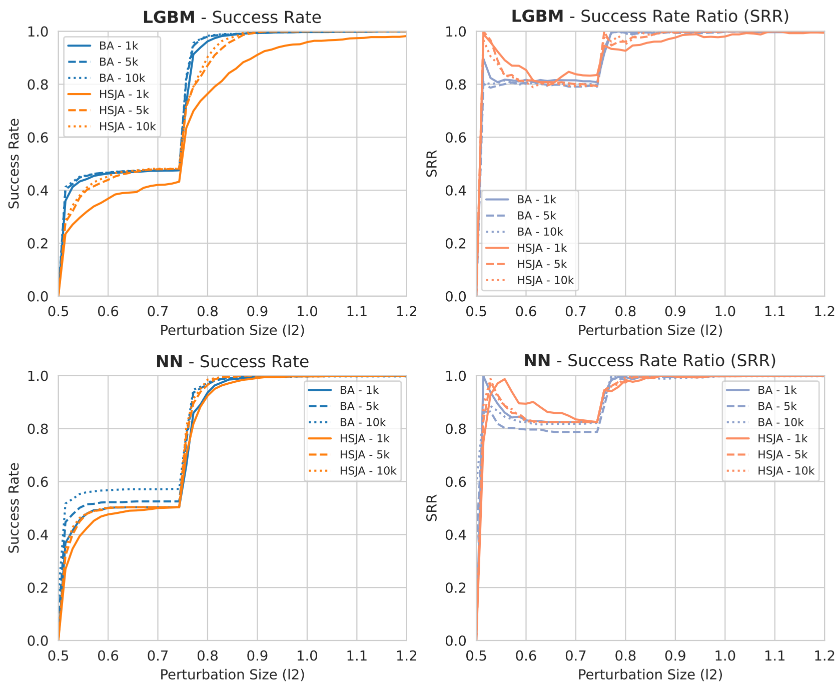


Figure 6.4: Applying adversarial attacks on BAF Base. Evaluating robustness regarding performance and fairness by measuring the success rate. The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column of plots displays the success rate for each perturbation size constraint, fixing a number of max queries constraint (1,000; 5,000; 10,000), for the Boundary Attack(BA) and the HopSkipJump(HSJA) Attack. The second column of plots shows a comparison of the Success Rate Ratio for the two attacks, with the same query constraints.

As for the Success Rate Ratio (SRR), both models exhibit similar behavior with the two methods. Following the initial success rate spike (reaching around 50%), the ratio also experiences a spike and stabilizes at around 0.80. This pattern indicates that for perturbation norms below l_2=0.75, the model demonstrates greater robustness for one group compared to the other. Subsequently, after the second spike, occurring near the l_2=0.75 threshold, the ratio approaches 1.0, implying equal robustness for perturbations of this size.

6.2.3 ACS Income

The third and last dataset is the dataset commonly used in the literature, ACS Income. It does not feature a distribution shift from split to split but has bias associated with the sensitive feature. In Figure 6.5, we can observe the results regarding the perturbation size of adversarial attacks against models trained in this dataset. Both models exhibit higher vulnerability to the Boundary Attack than to the HopSkipJump Attack with fewer queries, specifically under 3,000 queries for LGBM and under 4,000 queries for NN. However, their resilience against the attacks shifts as the number of queries

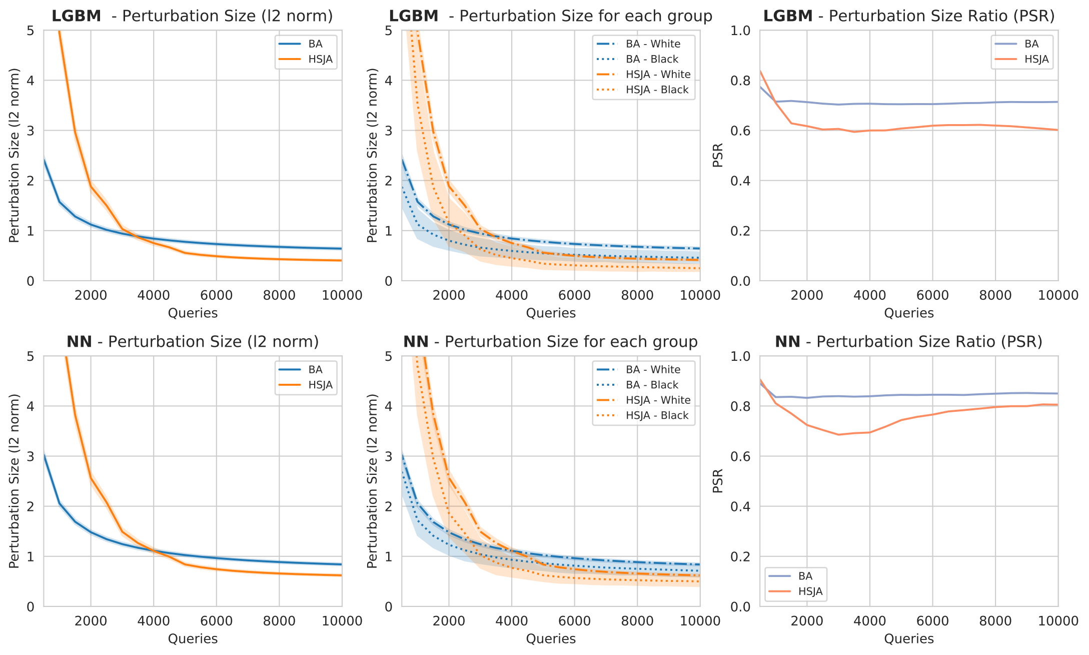


Figure 6.5: Applying adversarial attacks on ACS Income. Evaluating robustness regarding performance and fairness by measuring perturbation size (l_2 norm). The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column plots represent a comparison between the Boundary Attack (BA) and the HopSkipJump Attack (HSJA)'s perturbation norms, the second column separates the previous plots per sensitive group and the third plot shows a comparison of the Perturbation Size Ratio for the two attacks.

grows, becoming more vulnerable to the HopSkipJump Attack. Notably, both attacks converge within the range of [0.5, 1], with the HopSkipJump Attack outperforming for the two models. When we generate separate $l_2(queries)$ plots for each group, some differences become apparent, with the Black group (African Americans) exhibiting wider error margins, and the l_2 norms being notably lower. Verifying this observation, the Perturbation Size Ratio (PSR) reinforces the distinction between groups. LGBM emerges as the less fair model, with PSR figures hovering at approximately 0.7 for the Boundary Attack and approximately 0.6 for the HopSkipJump Attack. Conversely, the neural networks present higher PSR values, with the Boundary Attack's values slightly surpassing 0.80 and the HopSkipJump Attack's PSR oscillating between 0.7 and 0.8.

Focusing on the success rate, as shown in Figure 6.6, we once again, observe distinctive spikes in success rates at various perturbation norm values. Given the dataset's composition, featuring multiple categorical features, these spikes may be attributed to changes in these variables' categories. Notably, for both models, it becomes evident that 1,000 queries are insufficient for the HopSkipJump Attack to reach a 1.0 success rate (without using a l_2 norm constraint that is way looser than the ones we test in this experiment). When we consider the Success Rate Ratio (SRR), the models exhibit their lowest fairness when subjected to the HopSkipJump Attack with 1,000 queries, with values consistently below 0.8. This observation suggests different susceptibilities to attacks among different groups. In the remaining scenarios, prior to the initial spike, the SRR fluctuates between 0.6 and 0.8, marking the phase where the model's fairness in terms of robustness is smallest. However, after each spike, the SRR progressively converges to 1.0, indicating a progression towards a state of equal robustness for both groups.

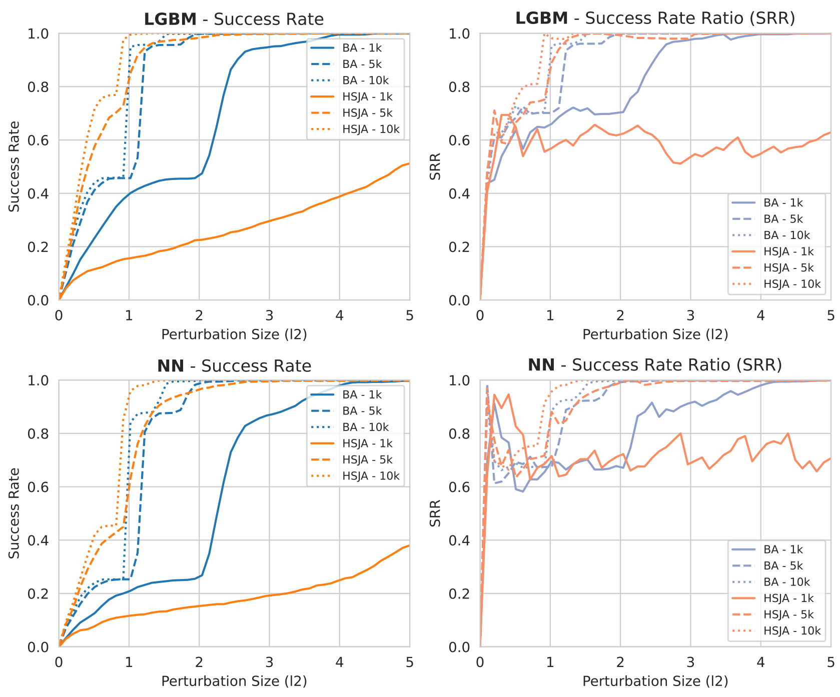


Figure 6.6: Applying adversarial attacks on ACS Income. Evaluating robustness regarding performance and fairness by measuring the success rate. The plots on the first row are relative to a LightGBM model and the ones on the second row to a Neural Network. The first column of plots displays the success rate for each perturbation size constraint, fixing a number of max queries constraint (1,000; 5,000; 10,000), for the Boundary Attack(BA) and the HopSkipJump(HSJA) Attack. The second column of plots shows a comparison of the Success Rate Ratio for the two attacks, with the same query constraints.

6.3 Conclusion

While the l_2 norm might not be entirely practical for assessing real-world robustness, our framework serves well in comparing robustness across models and against diverse attacks. We noticed that identical models exhibit varying levels of robustness when trained on different datasets, facing the same attack approaches. Additionally, our framework revealed that some models have higher robustness for some groups than for others, exposing the presence of robustness bias in some of the settings we studied. Finally, in the datasets with the same feature space (the BAF datasets), the perturbation sizes were higher in the setting where there is distribution shift between sets.

Chapter 7

Enhancing Model Robustness

In this thesis, we have proposed a perturbation taxonomy and a framework to evaluate model robustness, taking both fairness and performance into consideration. Thus far, we simply evaluated the robustness of models trained in three different datasets without any robustness concerns. In this chapter, we develop a model that takes robustness into account and test its resilience.

This section emphasizes the motivation behind building robust models and describes the methodology used to develop this kind of model in our work. We present results obtained during the training process and the model's robustness against the perturbations in our proposed taxonomy.

7.1 Motivation

A recurring topic in this work is the dynamics associated with the real world, where the conditions of the *development* phase of machine learning models differ from the conditions in *production*, as data distribution is usually not the same for diverse reasons. Examples of causes of those distributions are the perturbations we listed in the taxonomy we proposed in Chapter 3. Additionally, we have tested the impact of these perturbations in models trained without any robustness considerations, exposing their lack thereof.

The value of machine learning models lies in their ability to perform well in *production*. For example, if a bank deploys a fraud detection model, it wants it to correctly predict fraudulent instances, without discriminating against any sensitive group. If the model doesn't behave as it should, the bank might suffer monetary losses or even wrongfully impact someone's life due to a simple demographic attribute. With this in mind, measures should be taken to avoid these scenarios.

A common method used to deal with changes in data distribution is to constantly retrain models with more recent data. However, this process is expensive and might not always be effective [84]. As an alternative, one can build robust models. The objective of a robust model is to maintain both fairness and performance levels from *development* to *production*. These models account for the existence of changes in the data during the *development* phase, aiming to reduce its impact on the model's correctness. This way, a model's behavior in production should be closer to the one expected from development.

Therefore, entities should be wary of methods for building robust models, preventing unwanted scenarios like the ones aforementioned.

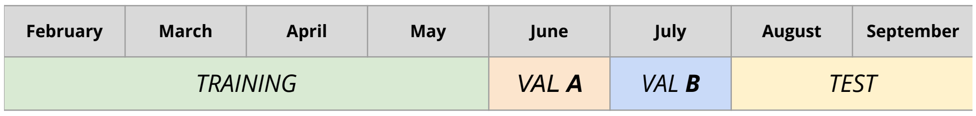
7.2 Methodology

The setting adopted for the development of robust models is the one that most closely simulates a real-world financial fraud scenario. The dataset used is the BAF Base, which intends to recreate a real bank account opening fraud detection dataset, and the model used is the LightGBM - as gradient-boosted trees models are a popular choice in this domain.

Concerning robust models, the existing literature heavily leans towards Adversarial Robustness, which consists of the ability of a model's resilience against adversarial attacks [85]. The literature provides numerous methods to increase adversarial robustness, and among them, adversarial training stands out as the most common [86, 87]. Adversarial training consists of crafting adversarial examples and incorporating them into the training process, exposing the model to these kinds of samples, so it can learn them. In our work, we choose to use this method for building robust models, as it is the most used approach.

7.2.1 Adversarial Training

We have seen that the BAF Base splits are made with time considerations. In this section, the splits are made differently, as represented in Figure 7.1. The validation set is now split into two parts, A and B. The adversarial training process uses the training set and validation set A, and the validation set B is used as an unseen set of data to tune the final threshold.



February	March	April	May	June	July	August	September
TRAINING				*VAL **A***	*VAL **B***	*TEST*	

Figure 7.1: Time splits for adversarial training.

Our implementation of adversarial training consists of three different phases: *attacking*, *evaluating*, and *training*. The process is described in Algorithm 1. In the *attacking phase*, we target a portion of the data and generate adversarial samples. The targetted samples consist only of true positives, as in fraud detection, an attacker wants to make the model misclassify a fraudulent instance (positive) as a legitimate one (negative). We target a portion of the true positives in the training set and all the true positives in the validation set, keeping a set of adversarial samples.

The *evaluation phase* uses validation set A, duplicating it into two sets: a *clean validation set* and an *adversarial validation set*. The first one consists of the original validation set, and the latter is the validation set but with the targetted samples replaced by their adversarial counterpart, in case the attack was successful. We evaluate the performance of the model in each of these datasets using the AUC. We choose this metric as it doesn't need any threshold that needs to be fitted in every adversarial round

- this way, we avoid using a threshold fitted in either of the two datasets that does not reflect an unbiased comparison between the two. Adversarial training stops whenever the adversarial AUC (measured on the *adversarial validation set*) is sufficiently close to the clean AUC (measured on the *clean validation set*). If that is not the case, the method proceeds to the next adversarial round, with retraining.

Finally, the incorporation of the generated adversarial samples occurs in the *training phase*. In this phase, the set of adversarial samples that are generated in the *attacking phase*, using both training and validation set, are added to the training data but flagged as fraudulent instances (positive). The model is then retrained with the new training set. Even though we use the AUC in the *evaluation phase*, we still need a threshold for making predictions in every adversarial round, as we need to query the model to generate adversarial attacks. This threshold is fitted on the *clean validation set* every round after retraining.

Algorithm 1 Adversarial Training

```
while 1 do
    ▷ attacking
        $adv_{train} \leftarrow attack(x_{train}, model, adv\_frac)$        ▷ generate adversarial samples from training
        $adv_{val} \leftarrow attack(x_{valA}, model, adv\_frac)$        ▷ generate adversarial samples from validation
    ▷ evaluating
        $x_{val\_adv} \leftarrow replace(x_{valA}, adv_{val})$        ▷ replace samples with successful adversarial counterparts
        $x_{val\_clean} \leftarrow x_{valA}$
        $AUC_{adv} \leftarrow compute\_metrics(x_{val\_adv}, model)$
        $AUC_{clean} \leftarrow compute\_metrics(x_{val\_clean}, model)$
      if $AUC_{clean} - AUC_{adv} > \epsilon$ then        ▷ $\epsilon$ is a tuneable threshold
        $break$
      end if
    ▷ training
        $\tilde{x}_{train} = x_{train} + as\_fraud(adv_{train}, adv_{val})$        ▷ add successful attacks to train as fraud
        $\widetilde{model} = fit\_model(\tilde{x}_{train}, model)$
        $threshold \leftarrow fit\_threshold(x_{val\_clean}, \widetilde{model})$
end while
    ▷ final training
        $\tilde{x}_{train} = \tilde{x}_{train} + x_{valA}$        ▷ Concatenate train and validation A
        $final\_model = fit\_model(\tilde{x}_{train}, \widetilde{model})$
        $threshold \leftarrow fit\_threshold(x_{valB}, final\_model)$        ▷ fit the threshold in unseen data
```

When the training converges (adversarial AUC is sufficiently close to the clean AUC), the validation set A is added to the train set, and the model is retrained a final time, using this concatenated data. The validation set B is then used to tune the final threshold of the model.

HopSkipJump Attack is the attack chosen to generate the adversarial samples used for training. The reasoning behind this choice is that even though Boundary Attack is more effective against the LightGBM in this dataset, the difference is negligible, making this attack more suitable for a process that involves various rounds due to its better time complexity. Furthermore, we choose a perturbation size constraint of $l_2 \leq 0.8$ and a max queries constraint of 5,000 queries, to limit the success rate. These values are chosen to take into account the results obtained in Chapter 6: the norm is high enough for the success rate to be 100% in the first adversarial round and the number of queries is chosen at a point where the method has already converged.

During the training process, model hyperparameters are kept the same as the ones obtained for the model in the previous section in every adversarial round. We consider that the results wouldn't vary too much if we were to perform hyperparameter tuning every adversarial round, since a large portion of the data is still the same data used during the previous training process. Additionally, the fraction of the true positives of the training set targetted is set to 5% and the threshold set for the adversarial training stopping is set to $\epsilon = 0.01$

7.3 Training a Model to be Robust

Throughout the adversarial training process, we closely monitored the progression of the AUC for both the *clean* and *adversarial* validation sets. Additionally, we evaluated the success rate of adversarial attacks on the validation set. The evolution of these magnitudes is represented in Figure 7.2. As anticipated, a trade-off occurs when sacrificing some performance on the clean validation, allowing us to enhance robustness. This is backed by the improvement of the performance on the adversarial validation set, transitioning from below AUC=0.5 to approximately AUC=0.73. Furthermore, the boost in adversarial robustness is demonstrated by the sharp decline in the success rate, approaching nearly 0%. This indicates our inability to launch successful attacks within the established constraints.

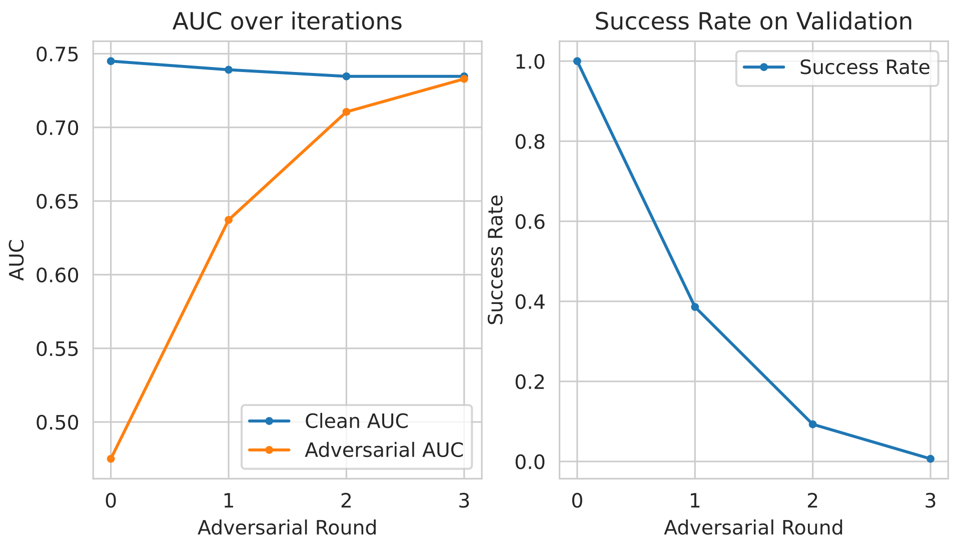


Figure 7.2: AUC comparison on clean and adversarial validation sets and Success Rate of attacks on the validation set.

Another noteworthy observation is the progressive increase in the FPR ratio during the training process, as depicted in Figure 7.3. Initially, LGBM's fairness in this dataset is at a lower level (approximately 31% in the test set). However, the adversarial training process successfully raises the FPR ratio to around 46%. Adversarial training has been shown to affect performance differently for different classes, and, in this case, it leads to a fairer model [66].

The model with enhanced robustness, denoted as the 'Robust LGBM', is evaluated on the test set, with the results available in Table 7.1. As expected, there was a modest drop in performance, with TPR

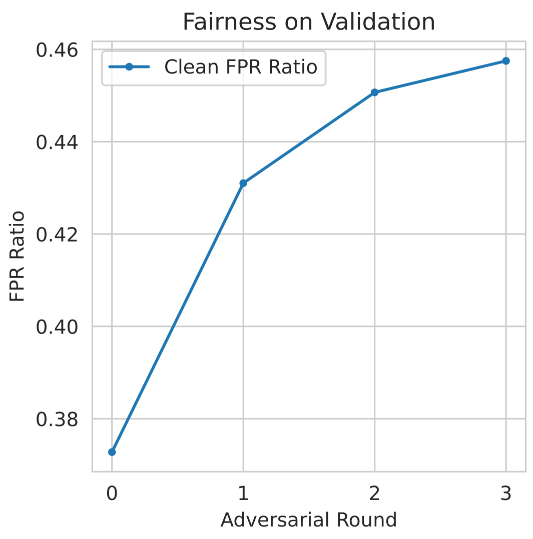


Figure 7.3: FPR Ratio evolution over the adversarial rounds.

decreasing from 56.50% to 54.66% when compared to the LGBM trained without robustness considerations. However, fairness metrics exhibited a substantial rise, surging from 31.30% to 59.09%, marking an even more substantial increase than observed on the validation set.

Table 7.1: Performance and Fairness comparison between the LGBM models trained with and without robustness concerns.

TPR		FPR Ratio	
LGBM	Robust LGBM	LGBM	Robust LGBM
56.50%	54.66%	31.30%	**59.09%**

In summary, with a slight compromise in performance, we nearly doubled fairness. Regarding the model's robustness, it is evaluated in the following section.

7.4 Testing the Robust Model's resilience

To enhance model robustness against adversarial attacks, we undertook adversarial training. To assess the success of this process, we now turn to evaluating the model's resilience against the very same perturbation. We leverage the framework introduced in the preceding chapters and, given that our primary aim was to boost the model's adversarial robustness, we focus solely on its evaluation in this specific context, as robustness is intricately tied to the nature of the perturbation. Even though the adversarial training process used the HopSkipJump Attack to generate the adversarial samples that were added to the training set, we also test the model against the Boundary Attack, given that it is still an adversarial attack.

In this instance, instead of contrasting lightGBMs with neural networks, our focus is directed towards comparing the original LGBM model trained on the BAF Base dataset without robustness considerations with the novel robust model. Results regarding perturbation size and PSR are shown in Figure 7.4.

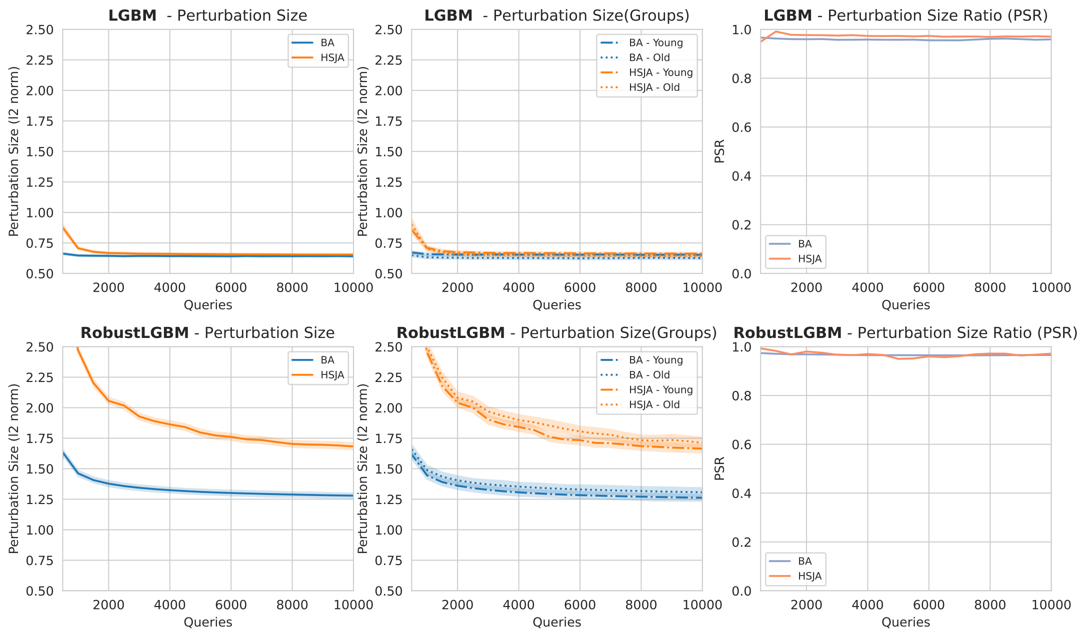


Figure 7.4: Comparing Advarial Robustness of a Robust and Robustness Blind Model. Evaluating robustness regarding performance and fairness by measuring the success rate. The plots on the first row are relative to a LightGBM model and the ones on the second row to a RobustLGBM. The first column plots represent a comparison between the Boundary Attack (BA) and the HopSkipJump Attack (HSJA)'s perturbation norms, the second column separates the previous plots per sensitive group and the third plot shows a comparison of the Perturbation Size Ratio for the two attacks.

The results regarding the LGBM are the same as we presented in the previous chapter: the model is slightly more robust against the HopSkipJump Attack, as the attacks have higher overall l_2 norms and the PSR is close to 1.0, suggesting that the model is equally as robust for both groups.

Shifting our focus to the RobustLGBM, we observe that fairness, as assessed through the PSR, closely aligns with its predecessor, maintaining proximity to 1.0. Further investigating fairness, the RobustLGBM showcases similar robustness across old and young demographics, irrespective of the attack type, with a slight inclination favoring the elderly. Notably, the l_2 norms of the attacks against the RobustLGBM are higher than those of the standard LGBM by over twofold, underscoring its heightened robustness. Additionally, despite experiencing the most substantial increase in resilience against the HopSkipJump Attack, its capacity to withstand Boundary Attacks has also improved.

As for the success rate, the results are depicted in Figure 7.5. The LGBM's outcomes are the same as the ones in the preceding chapter: a surge in success rate at around l_2=0.75, followed by a steadfast 100% success rate for all attack-queries combinations with $l_2 \geq$ 1.5. In terms of SRR, it stabilizes at approximately 0.8 before the aforementioned spike and nears the 1.0 threshold thereafter. Turning our focus to the RobustLGBM, the evolution of success rates evolves at a notably slower pace. The HopSkipJump Attack with 1,000 queries, for instance, only registers a 50% success rate at l_2=2.5, with the remaining attack-query permutations only attaining a 100% success rate near this point. In the context of SRR, it undergoes a gradual convergence toward 1.0, and as it nears this critical value, the amplitude of fluctuations diminishes across all attack-query combinations.

In summary, the implementation of adversarial training has successfully boosted the model's re-

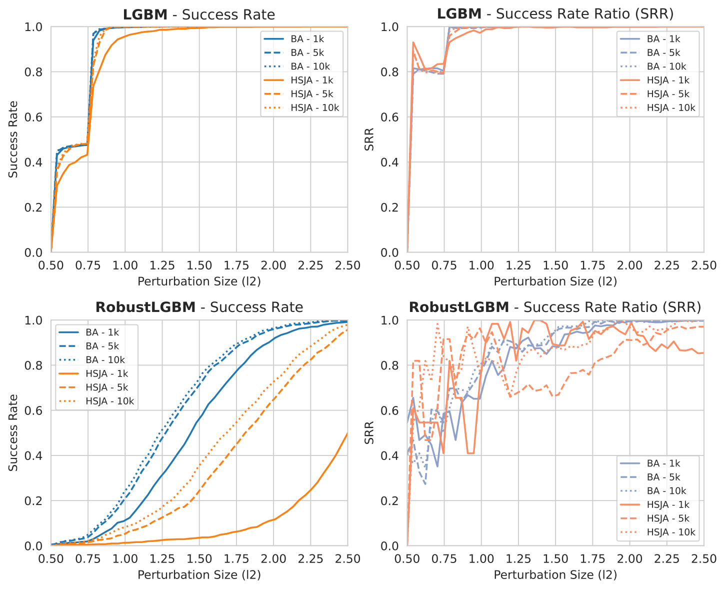


Figure 7.5: Comparing Advarial Robustness of a robust and robustness blind model. Evaluating robustness regarding performance and fairness by measuring the success rate. The plots on the first row are relative to a LightGBM model and the ones on the second row to a RobustLGBM. The first column of plots displays the success rate for each perturbation size constraint, fixing a number of max queries constraint (1,000; 5,000; 10,000), for the Boundary Attack(BA) and the HopSkipJump(HSJA) Attack. The second column of plots shows a comparison of the Success Rate Ratio for the two attacks, with the same query constraints.

silience against adversarial attacks, rendering it increasingly challenging to forge effective attack strategies. This augmentation in robustness is evidenced by the enlargement of perturbation sizes required for successful attacks and the discernibly slower convergence toward a 100% success rate. Concerning fairness indicators, the model exhibits consistency in PSR while yielding similar outcomes for SRR, although with a somewhat elevated level of variability.

7.5 Conclusion

In the context of the adversarial training process, as expected, we observed a slight performance decrease coupled with a significant boost in robustness. Adversarial training also contributed to an enhancement in fairness. Additionally, our framework effectively highlighted the disparity in adversarial robustness between the novel model and its original counterpart: as anticipated, the model explicitly trained for robustness exhibited markedly higher adversarial resilience. This underscores the utility of integrating our framework as a robustness test during the *development* phase.

Chapter 8

Conclusions and Future Work

The rapid advancements in machine learning (ML) have enabled artificial intelligence to enter a range of decision-making contexts. However, these algorithms are predominantly deployed in dynamic environments, featuring perturbations that can obstruct the model's intended functionality. In the realm of high-stakes decision-making, where errors exert a profound influence on individuals' lives, there arises a compelling need for comprehensive testing due to the need for higher guarantees that the model will behave as designed. Furthermore, within these critical scenarios, prioritizing fairness becomes paramount, driven by a concern to prevent adverse repercussions on individuals' lives due to sensitive attributes. Hence, there is a pressing need to build robust models that uphold values of fairness and performance even in the presence of these perturbations.

Building robust models stands as a challenge that has garnered significant attention within the current ML landscape. The prevailing perspective on robustness, as often highlighted in the literature, focuses on the importance of models demonstrating resilience against adversaries who purposely craft data samples in an attempt to deceive them – a quality referred to as adversarial robustness. Notably, a substantial proportion of research in this domain primarily concentrates on image data, whereas much of the broader spectrum of ML applications relies on tabular data. In the limited studies that do venture into this data domain, the incorporation of fairness considerations remains scarce, even within high-stakes scenarios.

Our endeavor was to establish a fresh perspective on robustness, extending its scope beyond adversarial robustness and encompassing fairness as a fundamental component. We argued that robustness should be tailored to address distinct forms of perturbations. We thus proposed a perturbation taxonomy that delineates various common challenges that ML models must effectively navigate to maintain correctness. In addition to adversarial attacks, our taxonomy incorporates a range of data issues, as they regularly pose unfavorable conditions that can lead to compromises in both fairness and performance. Our concept of robustness stresses the distinction between a model's resilience against adversarial attacks and its resilience against data issues, with each aspect warranting separate evaluation.

Furthermore, we introduced a framework that aligns with our robustness perspective, complementing the evaluation of models. This approach allows robustness to become a key consideration in the lifecycle of ML models in real-world applications. Running our fraud detection scenario, and drawing upon a standard setting frequently found in the literature, we used an empirical analysis resorting to our framework to unveil the vulnerability of popular models that were constructed without robustness concerns. These settings are distinctive as they allow for testing with tabular data, particularly in a severely imbalanced context, and notably in high-stakes decision-making environments where fairness considerations are imperative. Additionally, we tested training with a focus on adversarial robustness with promising results when compared with standard model training.

Our main contribution can be outlined as follows:

- We proposed a perturbation taxonomy that extends beyond the common topic of adversarial attacks, including different data issues that also pose problems to ML models.
- We developed a framework able to evaluate model robustness against the perturbations we mentioned during the development process.
- We evaluated the robustness of various models in different scenarios using tabular data, resorting to our framework. This process revealed vulnerabilities of the models when subjected to the different kinds of perturbations.
- We trained a model to be adversarially robust and leveraged our framework to draw a comparative analysis with a model that was trained without accounting for robustness. This exemplifies the utility of our proposition by enabling model comparisons during their development stages.

8.1 Future Work

For future work, we aim to broaden the perturbation taxonomy to include a wider array of challenges. In our current proposition, we have incorporated a selection of the most prevalent perturbations encountered in the context of fraud detection. However, it is evident that numerous other perturbations are worthy of study. For instance, we could introduce intangible perturbations, such as distribution shifts resulting from natural occurrences, like shifts in consumer behavior.

In the realm of dynamic environments, another challenge that models can grapple with is dynamic biases, i.e., that the bias embedded in the data used for model training diverges from what is encountered in the *production* phase. For instance, one sensitive group may exhibit a higher prevalence during training, only to exhibit the same prevalence post-deployment. Conversely, a scenario might arise where a specific group benefits from a certain feature-space relationship, granting them a predictive advantage during the development period, affecting the learning process, only to witness the disappearance of this advantage after deployment. Such challenges have the potential to impact fairness, rendering them a captivating subject for future exploration

We could also extend the taxonomy of perturbations in the topics of data issues and adversarial attacks. When it comes to data issues, we have the option to shift our focus toward label noise instead of altering features, as we have done so far. This expansion leads us to consider categories like Noise

Completely at Random, Noise at Random, and Noise not at Random. These categories involve introducing noise in distinct ways: symmetric noise, which is added independently of the true class and features; asymmetric noise, which depends on the true label; and noise that correlates with the feature values.

As for adversarial attacks, we can introduce a concept known as *poisoning attacks*. These attacks are designed to taint the training data, ultimately compromising the learning process. To study them, we can break down the setup into multiple timeframes. This involves sequential stages where we train the model, subject it to attacks, and then incorporate the successful attacks into the training set (without flagging them as fraudulent). These attacks could even be meticulously crafted to manipulate the decision boundary, making the model less fair. Such applications might, for instance, serve the purpose of discrediting government models. Furthermore, we could implement custom norms for tabular data, being able to simulate more closely real-world applications of our framework.

When developing robust models, our attention was primarily directed toward adversarial robustness. An alternative avenue we could explore in the future involves developing models that are resilient to other types of perturbations. This would allow us to test the effectiveness of our framework when comparing robustness for models that are trained to be robust against other perturbations.

Bibliography

[1] A. E. Khandani, A. J. Kim, and A. W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[2] A. Rajkomar, J. Dean, and I. Kohane. Machine learning in medicine. *New England Journal of Medicine*, 380(14):1347–1358, 2019.

[3] T. Brennan, W. Dieterich, and B. Ehret. Evaluating the predictive validity of the compas risk and needs assessment system. *Criminal Justice and behavior*, 36(1):21–40, 2009.

[4] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. In *Machine Learning: ECML-93: European Conference on Machine Learning Vienna, Austria, April 5–7, 1993 Proceedings 6*, pages 227–243. Springer, 1993.

[5] L. Yang and A. Shami. Iot data analytics in dynamic environments: From an automated machine learning perspective. *Engineering Applications of Artificial Intelligence*, 116:105366, 2022.

[6] A. Subbaswamy, R. Adams, and S. Saria. Evaluating model robustness and stability to dataset shift. In *International conference on artificial intelligence and statistics*, pages 2611–2619. PMLR, 2021.

[7] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[8] V. Ballet, X. Renard, J. Aigrain, T. Laugel, P. Frossard, and M. Detyniecki. Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*, 2019.

[9] J. Pombal, P. Saleiro, M. A. Figueiredo, and P. Bizarro. Prisoners of their own devices: How models induce data bias in performative prediction. *arXiv preprint arXiv:2206.13183*, 2022.

[10] I. H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[11] S. Barocas, M. Hardt, and A. Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[12] T. Calders, F. Kamiran, and M. Pechenizkiy. Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pages 13–18. IEEE, 2009.

[13] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[14] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[15] D. Biddle. *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Routledge, 2017.

[16] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[17] S. Mitchell, E. Potash, S. Barocas, A. D'Amour, and K. Lum. Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application*, 8:141–163, 2021.

[18] P. Gajane and M. Pechenizkiy. On formalizing fairness in prediction with machine learning. *arXiv preprint arXiv:1710.03184*, 2017.

[19] A. Cook and C. Glass. Above the glass ceiling: When are women and racial/ethnic minorities promoted to ceo? *Strategic Management Journal*, 35(7):1080–1089, 2014.

[20] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, pages 797–806, 2017.

[21] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[22] S. Corbett-Davies and S. Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*, 2018.

[23] J. Kleinberg. Inherent trade-offs in algorithmic fairness. In *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, pages 40–40, 2018.

[24] P. Saleiro, B. Kuester, L. Hinkson, J. London, A. Stevens, A. Anisfeld, K. T. Rodolfa, and R. Ghani. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*, 2018.

[25] H. Lamba, K. T. Rodolfa, and R. Ghani. An empirical comparison of bias reduction methods on real-world problems in high-stakes policy settings. *ACM SIGKDD Explorations Newsletter*, 23(1): 69–85, 2021.

[26] S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 329–338, 2019.

[27] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[28] F. Kamiran, T. Calders, and M. Pechenizkiy. Discrimination aware decision tree learning. In *2010 IEEE international conference on data mining*, pages 869–874. IEEE, 2010.

[29] D. Xu, S. Yuan, L. Zhang, and X. Wu. Fairgan+: Achieving fair data generation and classification through generative adversarial nets. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1401–1406. IEEE, 2019.

[30] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 35–50. Springer, 2012.

[31] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[32] T. Calders and S. Verwer. Three naive bayes approaches for discrimination-free classification. *Data mining and knowledge discovery*, 21(2):277–292, 2010.

[33] C. Dwork, N. Immorlica, A. T. Kalai, and M. Leiserson. Decoupled classifiers for group-fair and efficient machine learning. In *Conference on fairness, accountability and transparency*, pages 119–133. PMLR, 2018.

[34] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[35] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[36] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[37] R. Mangal, A. V. Nori, and A. Orso. Robustness of neural networks: A probabilistic and practical approach. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 93–96. IEEE, 2019.

[38] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[39] M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[40] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[41] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, 32(5):1179–1199, 2018.

[42] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1113–1118. IEEE, 2006.

[43] J. Perdomo, T. Zrnic, C. Mendler-Dünner, and M. Hardt. Performative prediction. In *International Conference on Machine Learning*, pages 7599–7609. PMLR, 2020.

[44] M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, pages 111–122, 2016.

[45] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[46] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[47] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[48] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[49] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[50] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[51] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[52] M. Cheng, S. Singh, P. Chen, P.-Y. Chen, S. Liu, and C.-J. Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. *arXiv preprint arXiv:1909.10773*, 2019.

[53] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[54] J. Chen, M. I. Jordan, and M. J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[55] J. Chen and Q. Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1739–1747, 2020.

[56] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[57] F. Cartella, O. Anunciacao, Y. Funabiki, D. Yamaguchi, T. Akishita, and O. Elshocht. Adversarial attacks for tabular data: Application to fraud detection and imbalanced data. *arXiv preprint arXiv:2101.08030*, 2021.

[58] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[59] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[60] H. Chen, H. Zhang, D. Boning, and C.-J. Hsieh. Robust decision trees against adversarial examples. In *International Conference on Machine Learning*, pages 1122–1131. PMLR, 2019.

[61] A. Rezaei, A. Liu, O. Memarrast, and B. D. Ziebart. Robust fairness under covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9419–9427, 2021.

[62] A. Mishler and N. Dalmasso. Fair when trained, unfair when deployed: Observable fairness measures are unstable in performative prediction settings. *arXiv preprint arXiv:2202.05049*, 2022.

[63] S. Milli, J. Miller, A. D. Dragan, and M. Hardt. The social cost of strategic classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 230–239, 2019.

[64] A. Estornell, S. Das, Y. Liu, and Y. Vorobeychik. Unfairness despite awareness: Group-fair classification with strategic agents. *arXiv preprint arXiv:2112.02746*, 2021.

[65] R. Fogliato, A. Chouldechova, and M. G'Sell. Fairness evaluation in presence of biased noisy labels. In *International Conference on Artificial Intelligence and Statistics*, pages 2325–2336. PMLR, 2020.

[66] H. Xu, X. Liu, Y. Li, A. Jain, and J. Tang. To be robust or to be fair: Towards fairness in adversarial training. In *International Conference on Machine Learning*, pages 11492–11501. PMLR, 2021.

[67] J. Ali, P. Lahoti, and K. P. Gummadi. Accounting for model uncertainty in algorithmic discrimination. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 336–345, 2021.

[68] V. Nanda, S. Dooley, S. Singla, S. Feizi, and J. P. Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 466–477, 2021.

[69] Y. Pruksachatkun, S. Krishna, J. Dhamala, R. Gupta, and K.-W. Chang. Does robustness improve fairness? approaching fairness with word substitution robustness methods for text classification. *arXiv preprint arXiv:2106.10826*, 2021.

[70] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[71] D. Solans, B. Biggio, and C. Castillo. Poisoning attacks on algorithmic fairness. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 162–177. Springer, 2020.

[72] S. Jesus, J. Pombal, D. Alves, A. Cruz, P. Saleiro, R. P. Ribeiro, J. Gama, and P. Bizarro. Turning the tables: Biased, imbalanced, dynamic tabular datasets for ml evaluation. *arXiv preprint arXiv:2211.13358*, 2022.

[73] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[74] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[75] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[76] R. Kohavi and B. Becker. Adult dataset, 1996. URL `https://archive.ics.uci.edu/ml/datasets/adult`.

[77] F. Ding, M. Hardt, J. Miller, and L. Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in neural information processing systems*, 34:6478–6490, 2021.

[78] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[79] J. Feng, Y. Yu, and Z.-H. Zhou. Multi-layered gradient boosting decision trees. *Advances in neural information processing systems*, 31, 2018.

[80] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[81] A. F. Cruz, P. Saleiro, C. Belém, C. Soares, and P. Bizarro. Promoting fairness through hyperparameter optimization. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1036–1041. IEEE, 2021.

[82] J. Kaiser. Dealing with missing values in data. *Journal Of Systems Integration (1804-2724)*, 5(1), 2014.

[83] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[84] Y. Wu, E. Dobriban, and S. Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR, 2020.

[85] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[86] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

[87] E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Appendix A

Experimental Setup

A.1 Hyperparameters

Table A.1: Hyperparameter grid for LightGBM.

Hyperparameters	Distribution	Values	Description
boosting_type	-	dart	Boosting type
enable_bundle	-	false	Enabling Exclusive Feature Bundling
n_estimators	uniform	{100, 1000}	Number of base tree learners
num_leaves	uniform	{10, 1000}	Maximum leaves of a tree
min_child_samples	log-uniform	{1, 500}	Min. num. samples needed to create a leaf node
learning_rate	uniform	[0.001, 0.1]	Boosting learning rate

Table A.2: Hyperparameter grid for Neural Network. The dimension and number of the hidden layers were sampled from a pool composed of the following combinations: the number of hidden layers varies from 1 to 4; the size of each hidden layer is a power of 2 in the range [2, 128].

Hyperparameters	Distribution	Values	Description
activation	-	{ReLU, LeakyReLU, Tanh}	Activation Function
learning_rate	uniform	[0.00001, 0.01]	Model learning rate
dropout_rate	uniform	[0.0, 0.1]	Dropout probaility
batch_size	-	{64, 128, 256, 512}	Batch size
batch_norm	-	{True, False}	Batch normalization
weight_decay	uniform	[0.0001, 0.01]	Adam weight decay
dim_layers	-	see caption	Number of neurons in each layer
n_epochs	log-uniform	1,5	Number of epochs
optimizer	-	*adam*	Optimizer
loss	-	Binary Cross Entropy(BCE)	Loss function

Appendix B

Data Issues Results

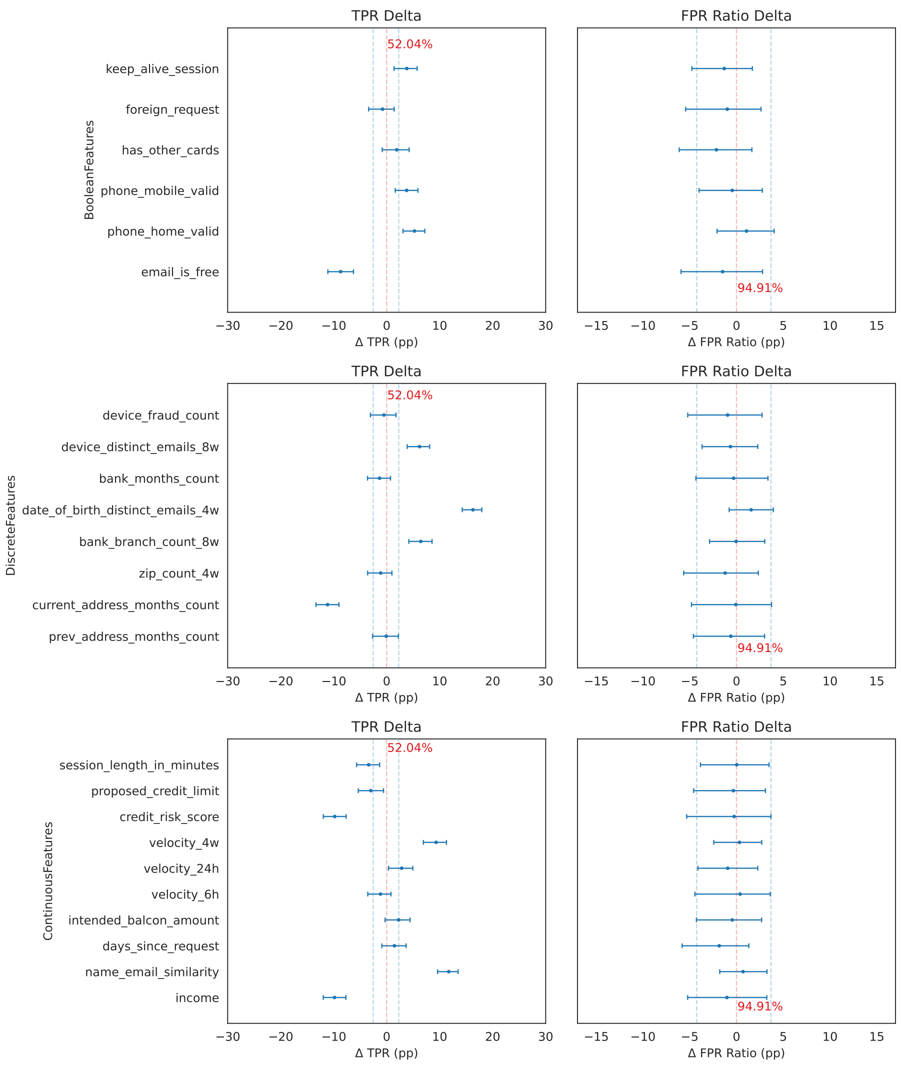


Figure B.1: Feature dropping using imputation replacement on LGBM trained on BAF IID.

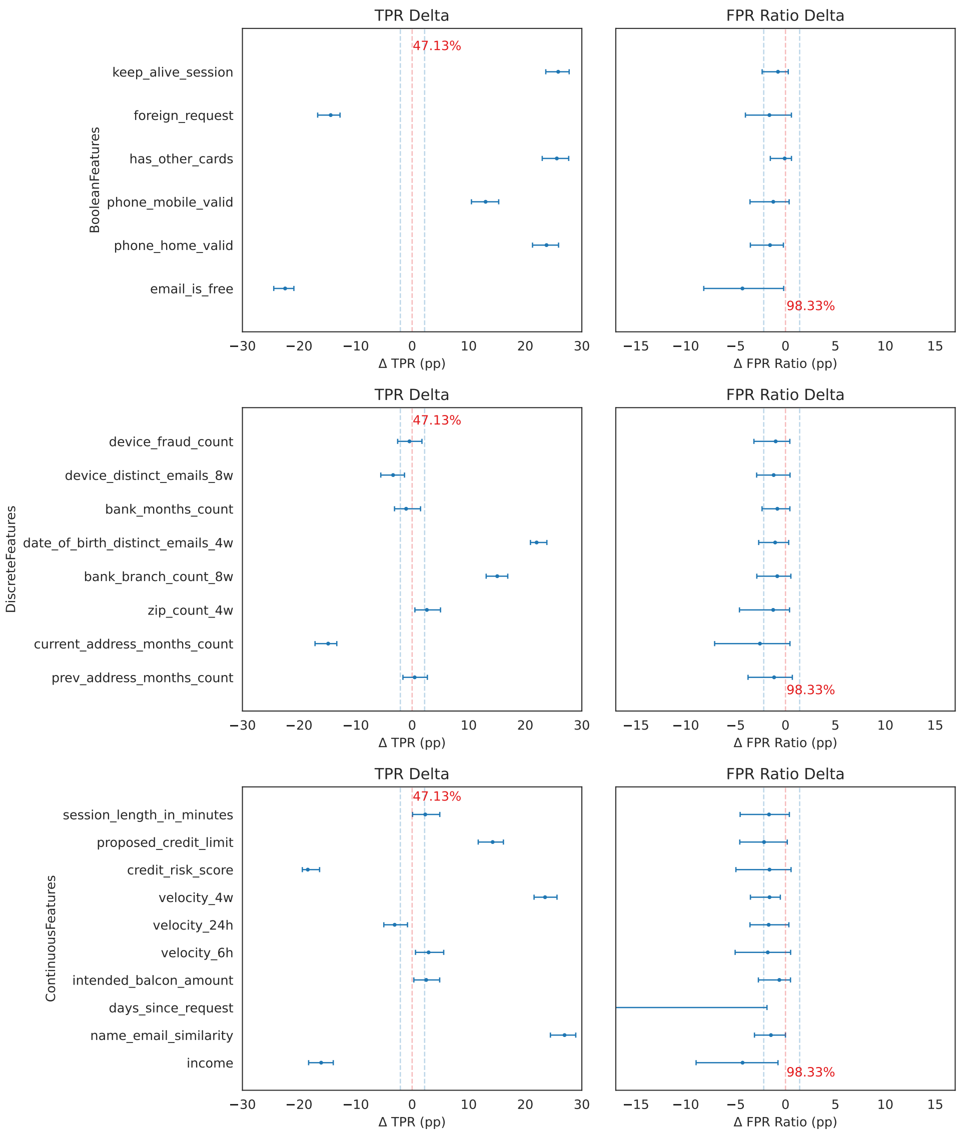


Figure B.2: Feature dropping using imputation replacement on NN trained on BAF IID.

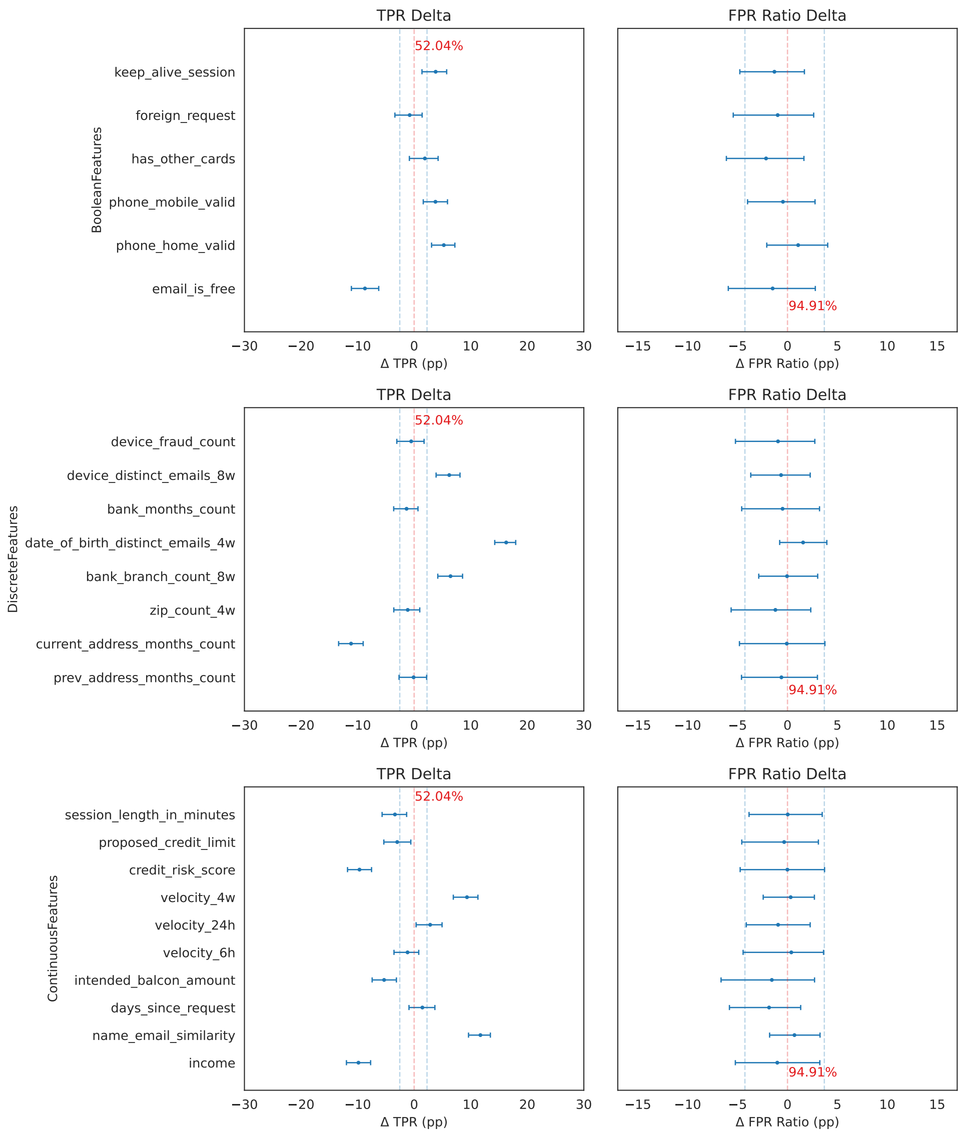


Figure B.3: Feature dropping using 0 replacement on LGBM trained on BAF IID.

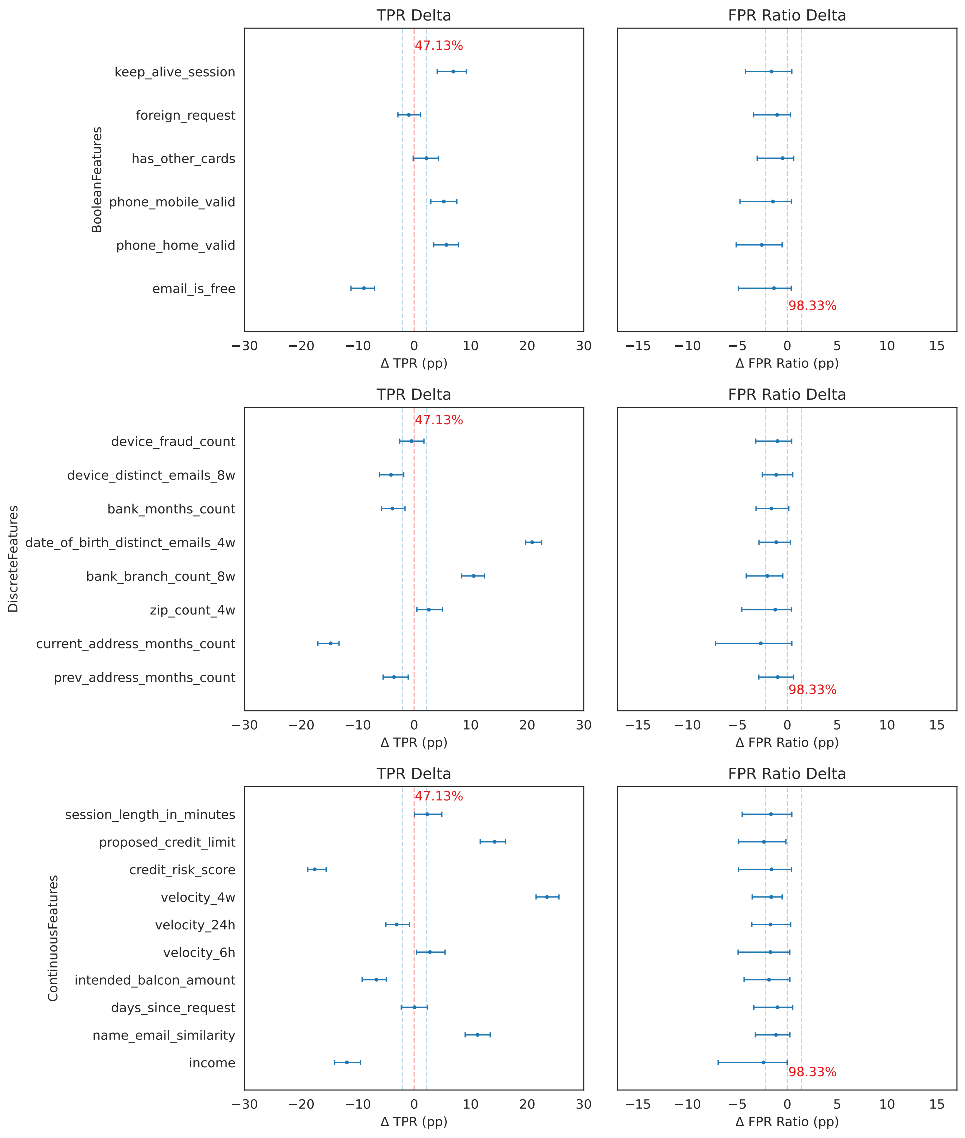


Figure B.4: Feature dropping using 0 replacement on NN trained on BAF IID.

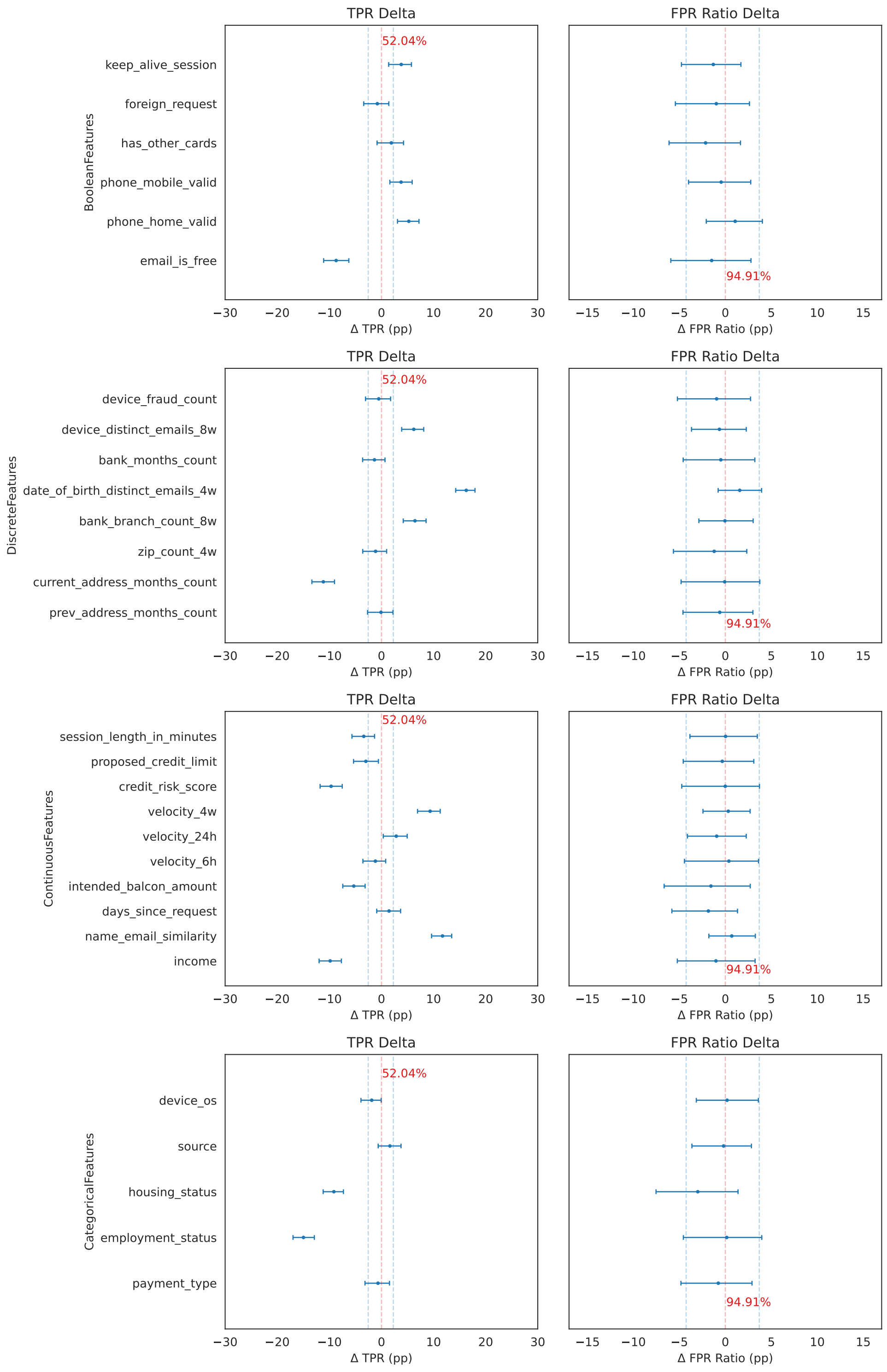


Figure B.5: Feature dropping using NaN replacement on LGBM trained on BAF IID.

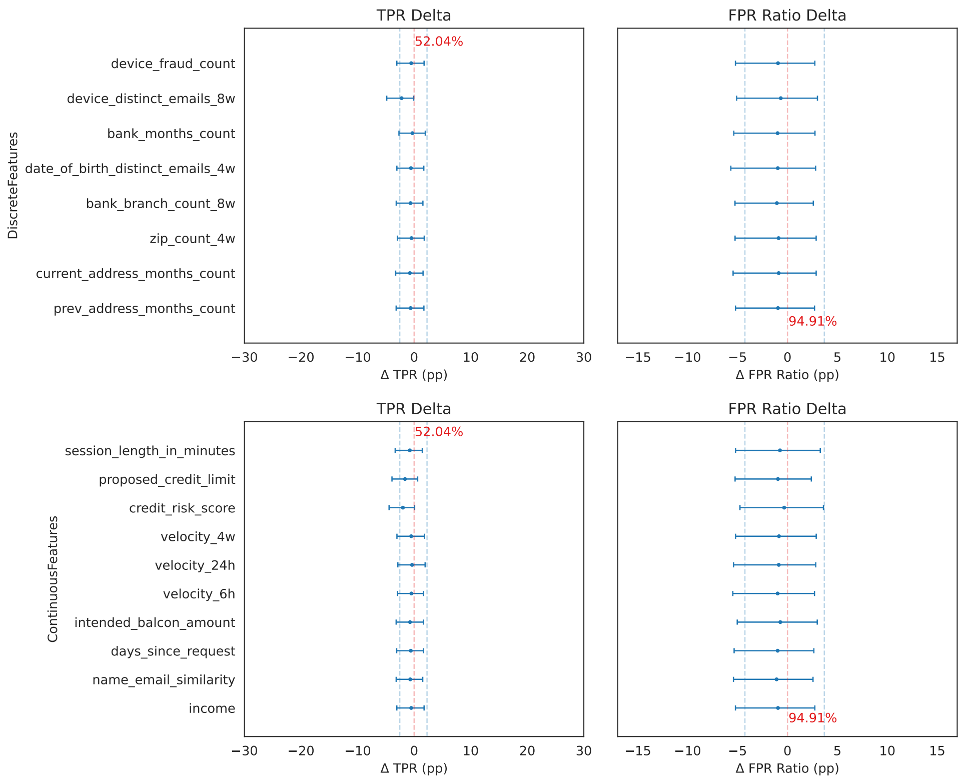


Figure B.6: Feature clipping using 5th and 95th percentiles as thresholds on LGBM trained on BAF IID.

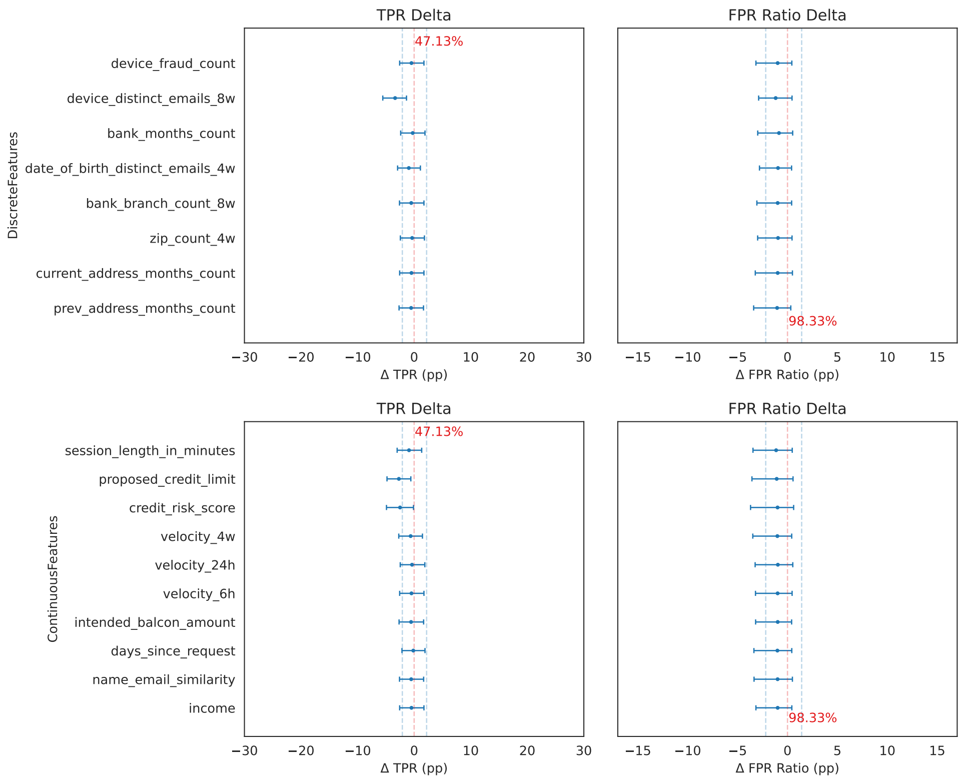


Figure B.7: Feature clipping using 5th and 95th percentiles as thresholds on NN trained on BAF IID.

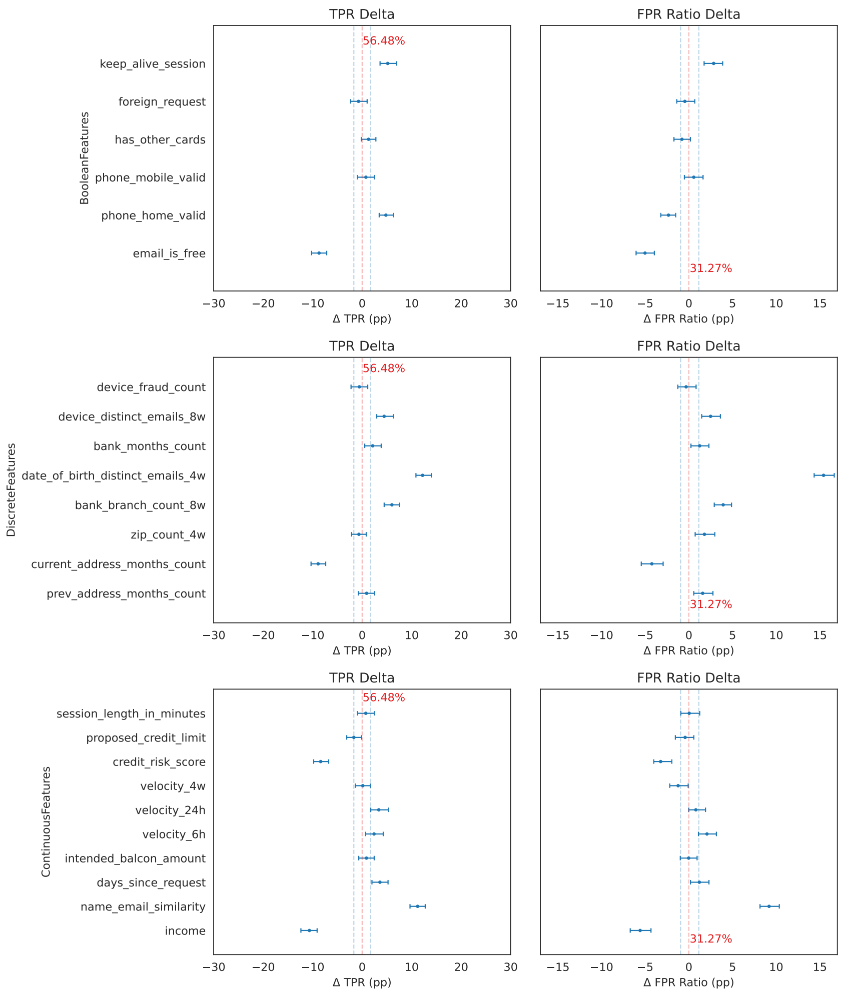


Figure B.8: Feature dropping using imputation replacement on LGBM trained on BAF Base.

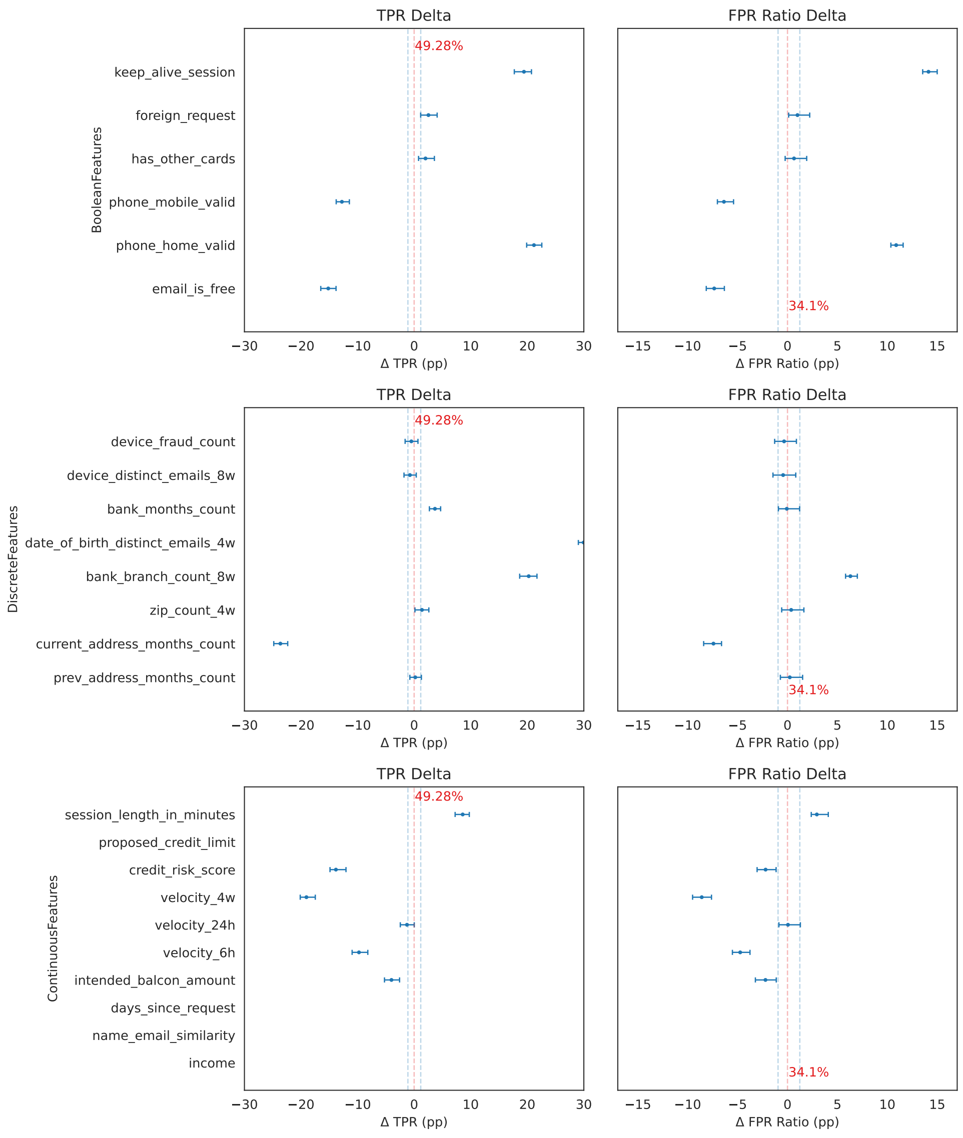


Figure B.9: Feature dropping using imputation replacement on NN trained on BAF Base.

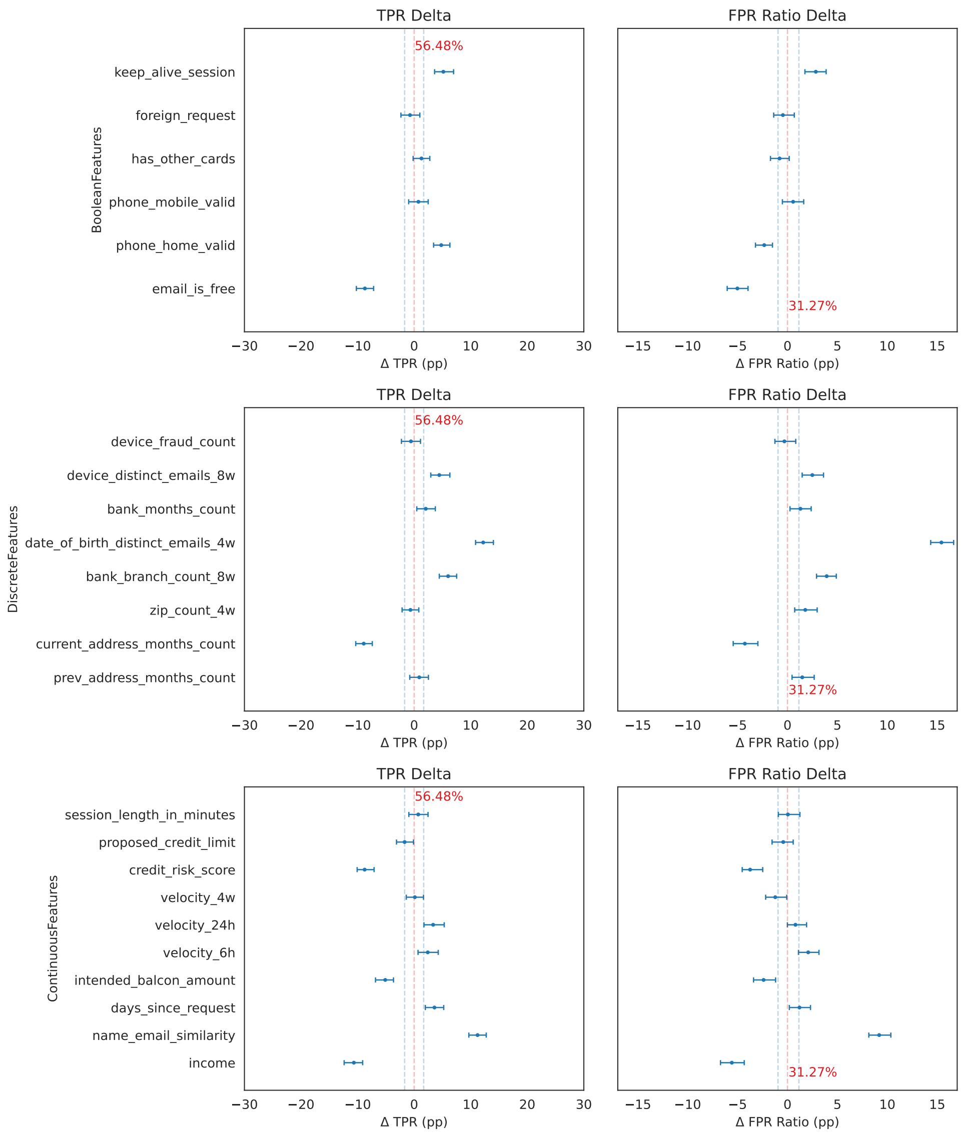


Figure B.10: Feature dropping using 0 replacement on LGBM trained on BAF Base.

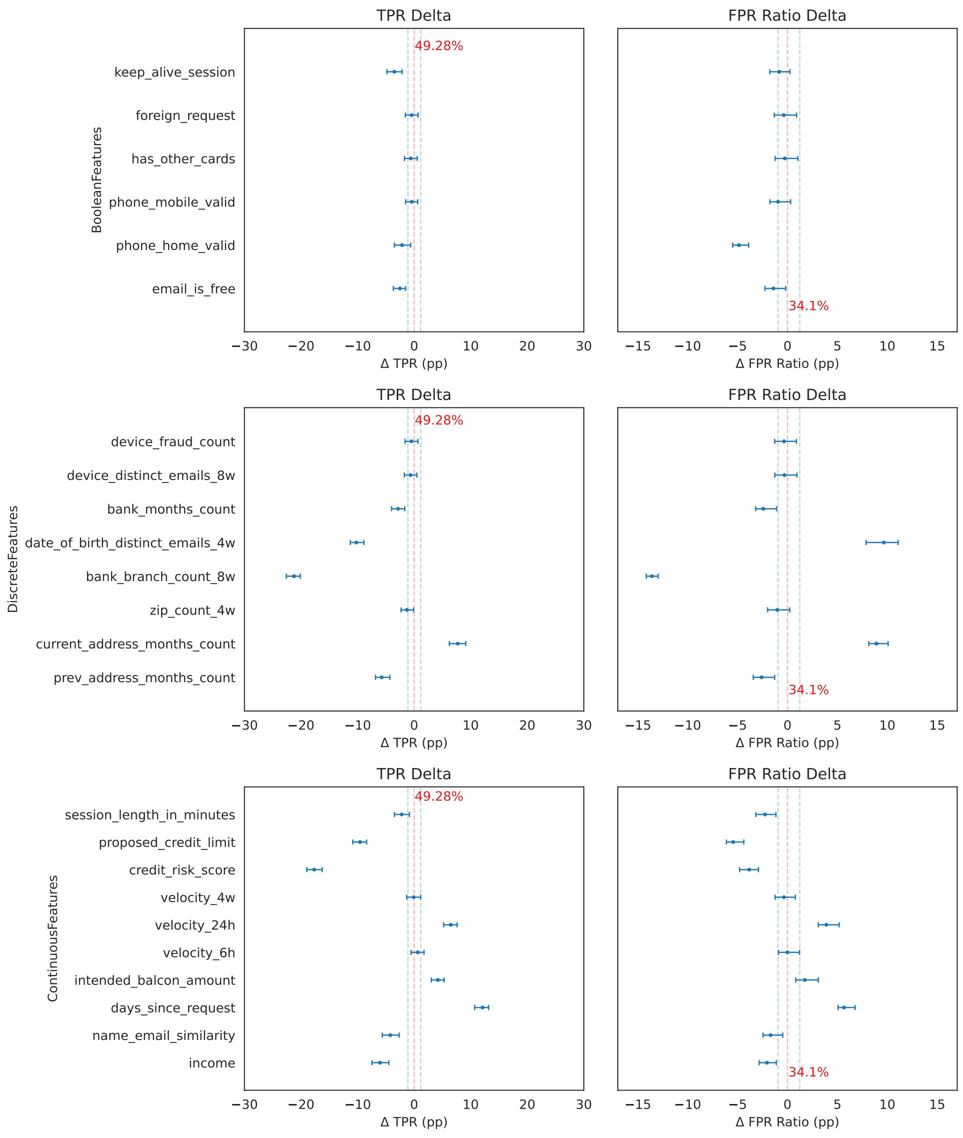


Figure B.11: Feature dropping using 0 replacement on NN trained on BAF Base.

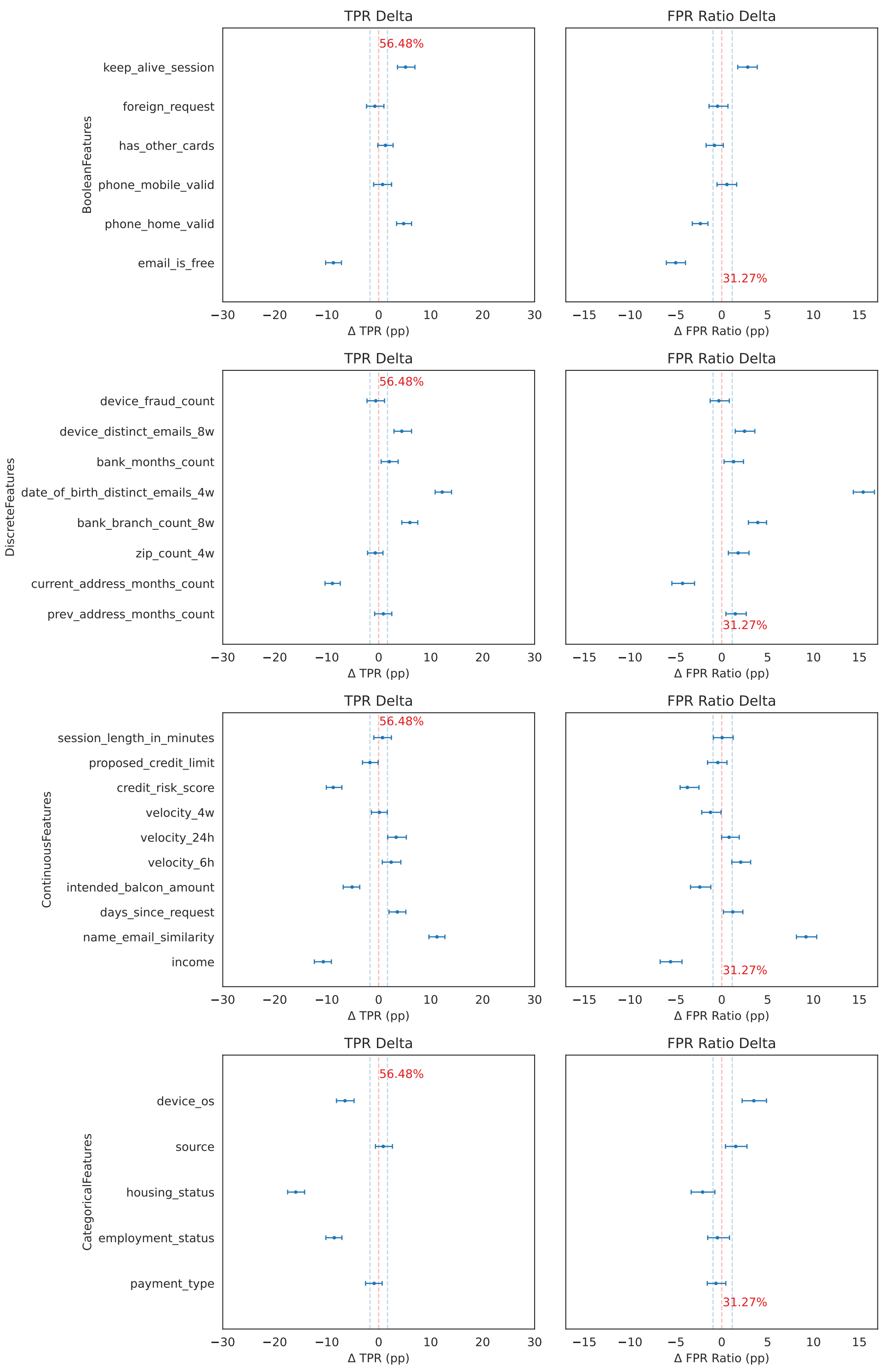


Figure B.12: Feature dropping using NaN replacement on LGBM trained on BAF Base.

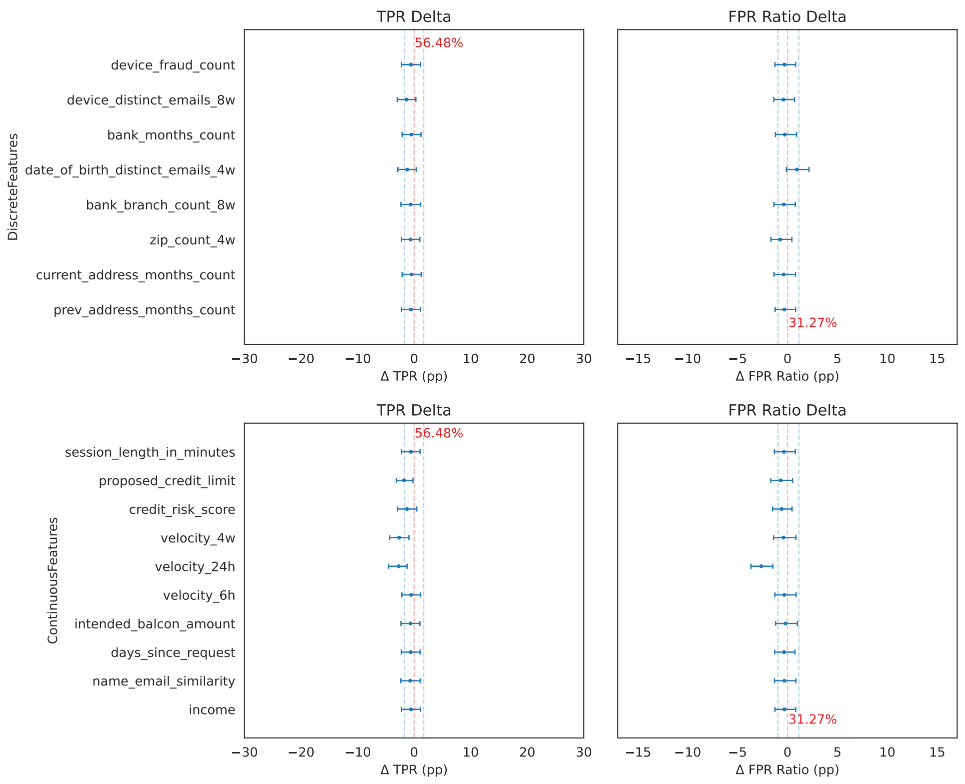


Figure B.13: Feature clipping using 5th and 95th percentiles as thresholds on LGBM trained on BAF Base.

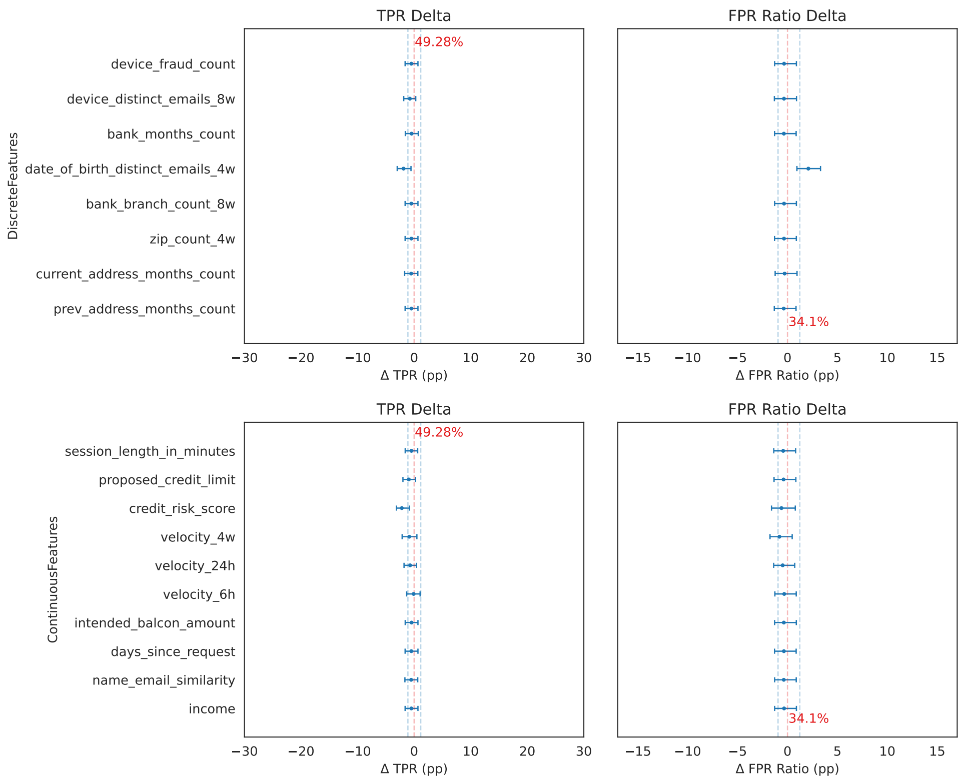


Figure B.14: Feature clipping using 5th and 95th percentiles as thresholds on NN trained on BAF Base.

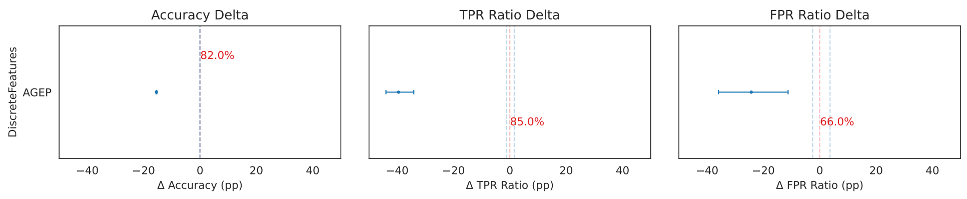


Figure B.15: Feature dropping using imputation replacement on LGBM trained on ACS Income.

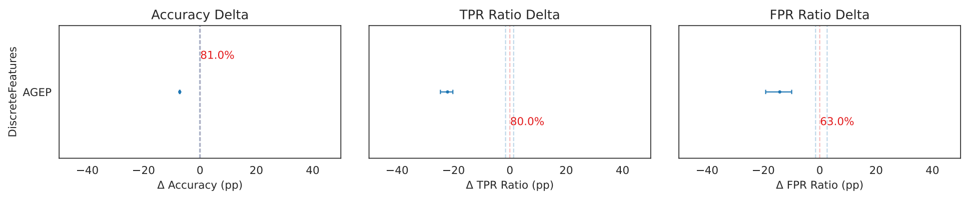


Figure B.16: Feature dropping using imputation replacement on NN trained on ACS Income.

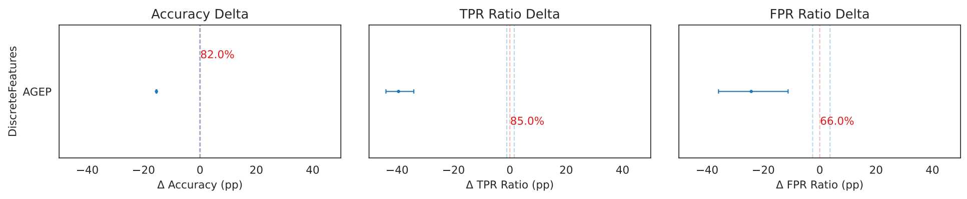


Figure B.17: Feature dropping using 0 replacement on LGBM trained on ACS Income.

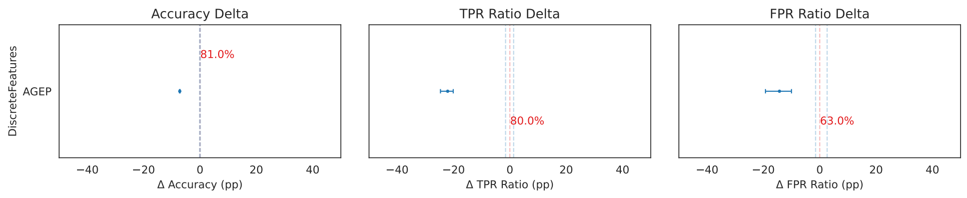


Figure B.18: Feature dropping using 0 replacement on NN trained on ACS Income.

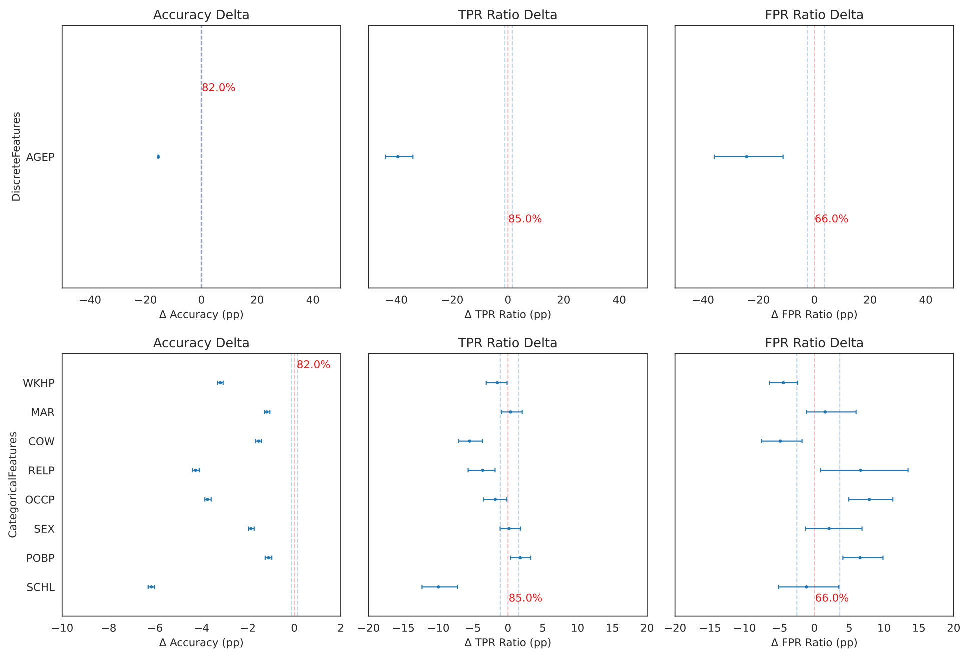


Figure B.19: Feature dropping using NaN replacement on LGBM trained on ACS Income.

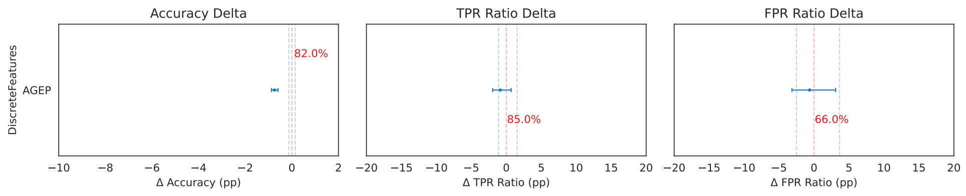


Figure B.20: Feature clipping using 5th and 95th percentiles as thresholds on LGBM trained on ACS Income.

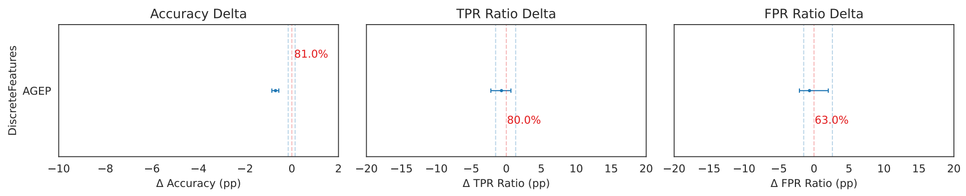


Figure B.21: Feature clipping using 5th and 95th percentiles as thresholds on NN trained on ACS Income.