All Actuaries Summit
Think bigger, live better
1-3 May 2024 • Gold Coast
Actuaries Institute.

Why You Should Not Trust Interpretations in Machine Learning

Prepared by Xi Xin, Giles Hooker, and Fei Huang

Presented to the Actuaries Institute
2024 All-Actuaries Summit
1-3 May 2024

Actuaries Institute
ABN 69 000 423 656
Level 2, 50 Carrington Street, Sydney NSW 2000
P +61 (0) 2 9239 6100 | actuaries.asn.au

Why You Should Not Trust Interpretations in Machine Learning: Adversarial Attacks on Partial Dependence Plots

Xi Xin[1], Giles Hooker[2], and Fei Huang[1]

[1] *UNSW Sydney, School of Risk and Actuarial Studies*

[2] *University of Pennsylvania, Wharton School Department of Statistics and Data Science*

Abstract

The adoption of artificial intelligence (AI) across industries has led to the widespread use of complex black-box models and interpretation tools. This paper proposes an adversarial framework to uncover the vulnerability of permutation-based interpretation methods for machine learning tasks, with a particular focus on partial dependence (PD) plots. This adversarial framework modifies the original black box model to manipulate model predictions for instances in the extrapolation domain, resulting in PD plots that can hide discriminatory behaviors while maintaining the prediction accuracy of the original model. This framework can produce multiple fooled PD plots via a single model. By using real-world datasets including an auto insurance claims dataset and COMPAS dataset, our results show that it is possible to intentionally hide the discriminatory behaviour of a predictor and make the black-box model appear neutral through interpretation tools like PD plots while retaining almost all the predictions of the original black-box model.

1 Introduction

In recent years, industries have increasingly adopted artificial intelligence (AI), leading to the widespread use of complex AI models. These AI applications can enhance efficiency and accuracy, resulting in time and cost savings. However, these models often operate as "black boxes", meaning while we can observe their inputs and outputs, their inner workings remain opaque. This lack of transparency has raised concerns from both regulators and consumers, especially when black-box AI models are used in critical decision-making scenarios.

In this paper, we propose an adversarial framework to demonstrate the susceptibility of partial dependence (PD) plots to adversarial attacks, specifically revealing how these plots can be manipulated by exploiting the extrapolation behavior of correlated features and the aggregation of heterogeneous effects during the averaging process. This adversarial framework modifies the original black box model to manipulate model predictions for instances in the extrapolation domain, which produces deceived PD plots that can hide discriminatory behaviors while maintaining the prediction accuracy of the original model. Empirical insurance and COMPAS datasets are applied to demonstrate the effectiveness of this model. Our results show that it is possible to intentionally hide the discriminatory behaviour of a predictor and make the black-box model appear neutral through interpretation tools like PD plots while retaining almost all the predictions of the original black-box model.

To gain insights into the relationships between model inputs and outputs, various interpretation methods from the growing field of interpretable machine learning can be potentially employed to support the application of black-box models. For example, some of these interpretation methods have garnered attention within sectors such as insurance (Kuo & Lupton, 2020; SOA, 2021; Delcaillau et al., 2022), credit scoring (Bücker et al., 2022; Szepannek & Lübke, 2023) or healthcare (Mohanty & Mishra, 2022) in recent years. These methods can be broadly categorized into three groups (SOA, 2021): 1) feature importance, 2) methods for understanding the relationships between model inputs and outputs (main effects), and 3) methods for identifying and visualizing interaction effects. PD plots are suggested as interpretation tools by regulators (NAIC, 2020) and practitioners (SOA, 2021) in the insurance industry. They are frequently employed in the literature when black-box insurance models are implemented or proposed – ranging from insurance (Guelman, 2012; Yang et al., 2018; Lee & Lin, 2018; Xie, 2021; Henckaerts et al., 2021) to customer churn (Lemmens & Croux, 2006; Matuszelański & Kopczewska, 2022), and criminal justice (Berk & Bleich, 2013).

The interpretability, explainability, and transparency of black-box algorithms are commonly discussed ethical challenges across various AI principle documents (Stanford HAI, 2019, p.149). Nevertheless, the definitions and interpretations of these principles tend to vary across scientific disciplines and among different stakeholders, including policymakers, technical standardization communities, legal scholars, and applied AI practitioners (Panigutti et al., 2023). Specifically, the EU's General Data Protection Regulation (GDPR) has been interpreted by some as containing a "right to explanation," a topic which continues to spark debate regarding the legal existence and the feasibility of such a right under the GDPR (Wachter et al., 2017; Bordt et al., 2022).

A recent regulatory development in the insurance domain is the expansion of the National Association of Insurance Commissioners' Predictive Models White Paper (NAIC, 2022) in the US insurance industry. It has extended its scope from primarily reviewing rate filings based on Generalized Linear Models (GLMs) (NAIC, 2020) to include tree-based models like random forests and gradient boosting machines. The White Paper provides state insurance regulators with comprehensive guidelines for assessing predictive models. Under these guidelines, insurers using tree-based methods in rate filings are expected to implement interpretability plots to describe the relationship between each predictor variable and the target variable, which could range from frequency and severity to loss costs and expenses. Insurers should offer rational explanations for the observed relationship, and are also recommended to obtain variable importance plots to identify significant variables influencing the model's outcomes.

Although stakeholders may possess a certain level of understanding regarding the limitations of these interpretation methods, there is often a lack of awareness regarding the inherent vulnerabilities of these methods. A growing body of literature emphasizes the need for caution in the use of these methods, as they can be unreliable and prone to provide misleading information. For instance, Rudin (2019) asserts that these explanations are fundamentally flawed as they can never be fully

faithful to the original model. Hooker et al. (2021) demonstrate the vulnerability of permutation-based interpretation methods due to the extrapolation behavior of correlated features. Molnar et al. (2022) uncover and review the general pitfalls of model-agnostic interpretation methods.

Recent years have seen studies developing adversarial attacks of certain interpretation methods, such as LIME, SHAP (Slack et al., 2020; Baniecki & Biecek, 2022; Laberge et al., 2022) and partial dependence plots (Baniecki et al., 2023). Building upon this line of research, we present the vulnerability of permutation-based interpretation methods via adversarial attacks on partial dependence (PD) plots – a widely utilized interpretation method. There is limited existing literature on developing an adversarial framework to fool PD plots. The only paper we could find is a recent study by Baniecki et al. (2023). Compared to their model which relies on a poisoned dataset to manipulate the PD plots, our framework modifies the model instead, which can achieve substantial fooling effectiveness and allow auditors (regulators) to examine the datasets used. Our model also extends Slack et al. (2020)'s scaffolded classifier to a scaffolded regressor framework. Notably, Slack et al. (2020) directly used the permuted data generated by LIME or kernel SHAP in their adversarial framework, however, our approach uses only extrapolated permuted PD data as compensating outputs. This subset, which falls within the extrapolation domain, is less likely to be observed in real data and represents only a portion of all permuted data generated for PD plots. This specificity enhances the efficiency and accuracy of the fooling process, as opposed to using the entire set of permuted data.

Our findings raise concerns about the use of permutation-based interpretation methods for machine learning tasks that require interpretation. This is because the discriminatory behavior of a predictor can be intentionally hidden by tools such as PD plots which make the model appear neutral while preserving nearly all original discriminatory predictions.

The rest of the paper is organized as follows. Section 2 reviews related studies. Section 3 introduces PD plots, discussing both their properties and limitations. Section 4 outlines our methodology for constructing an adversarial framework designed to manipulate PD plots. Section 5 examines the effectiveness of our framework using real-world datasets. Section 6 offers insights and recommendations for regulators and practitioners. Section 7 concludes the paper.

2 Related Studies

There are very few studies on adversarial attacks on global interpretation methods applied to tabular data, even though these methods are becoming increasingly popular in critical areas such as insurance and credit scoring. In a related work, Baniecki et al. (2023) approached fooling PD as an optimization problem using genetic and gradient algorithms, with adversarial data perturbations by supplying a poisoned dataset. Most adversarial attacks on interpretation methods involve either perturbing the data through small changes or modifying the model itself. Our adversarial framework can be viewed as a modification of the model and is distinctive for preserving the original model's

predictive performance while enabling auditors to examine the input dataset.

In a notable earlier study, Slack et al. (2020) demonstrated that local interpretation methods relying on input perturbations, such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017), can be vulnerable to adversarial attacks. These attacks exploit the use of synthetic neighborhood data points generated by LIME or SHAP, deviating from the underlying data distribution. Building upon this notion, our work expands on this idea by identifying and exploiting the extrapolated data generated by PD to deceive the PD itself via an adversarial framework. Our approach can be seen as an extension of Slack et al. (2020)'s scaffolded classifier to a scaffolded regressor framework, which serves as the inspiration for our methodology. Apart from these, other global or local interpretation methods for tabular data are susceptible to adversarial attacks, including global SHAP (Baniecki & Biecek, 2022; Laberge et al., 2022) and counterfactual explanations (Slack et al., 2021).

In the broader landscape of machine learning, the existing literature on adversarial attacks of interpretation tools extensively covers attacks targeting model-specific explanations. These studies primarily focus on deep neural networks applied to image data, as evidenced by works such as Ghorbani et al. (2019); Dombrowski et al. (2019); Heo et al. (2019); Dimanov et al. (2020), rather than the context of tabular data that we explore. Prior to these works, earlier studies primarily focused on adversarial attacks targeting model predictions (Szegedy et al., 2013; Goodfellow et al., 2014). Interestingly, adversarial examples were generated to deceive model classifications in Goodfellow et al. (2014) by altering pixel values to fall outside the training data distribution. This exploits the vulnerabilities of certain neural network architectures to "extrapolation" in image classification for conducting adversarial attacks. In comparison, we exploit the extrapolation vulnerabilities of PD plots – in a different context for permutation-based interpretation methods – to carry out adversarial attacks. For a comprehensive survey on adversarial attacks on model explanations, along with the corresponding defenses against such attacks, we direct readers to Baniecki & Biecek (2023).

3 Partial Dependence Plots

3.1 Notation and Definitions

Let y be the target variable, $\mathbf{x}$ be a set of predictor variables, and $\hat{f}$ be a trained predictive machine learning (ML) model, $\hat{\mathrm{y}} = \hat{f}(\mathbf{x})$. The feature matrix is denoted by X, and the target vector is $\mathbf{y}$. In this paper, the superscript (i) indicates an individual observation, and the subscript j refers to a specific featu: $x_j^{(i)}$ specifies the jth feature of the ith observation. Given the observed training data $\{X, \mathbf{y}\} = \{\mathbf{x}^{(i)}, \mathrm{y}^{(i)}\}_{i=1}^{n}$, $y^{(i)} \in \mathbb{R}$ and $\mathbf{x}^{(i)} \in \mathbb{R}^p$, where $n \in \mathbb{N}$ is the number of observations and $p \in \mathbb{N}$ is the number of features.

Partial Dependence (PD) Plot: The PD plot displays the marginal effect of a feature on the prediction (Friedman, 2001). Let X_S be a feature subset of X (typically $|S| = 1$ or 2), denoted as $X_S \subset \{X_1, \ldots, X_p\}$, and X_C be the complement subset, $X_S \cup X_C = X$. The PD function is defined

as

$$\mathrm{PD}_S(X_S) = \mathbb{E}_{X_C}[\hat{f}(X_S, X_C)] = \int \hat{f}(X_S, X_C) d\mathbb{P}(X_C) \tag{3.1}$$

When there is only one feature of interest X_j in S, let $\mathbf{x}_{-j}^{(i)}$ denote the i^{th} row of X without the j^{th} feature, the above can be estimated from the training sample by

$$\widehat{\mathrm{PD}}_j(x_j) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_j, \mathbf{x}_{-j}^{(i)}) \tag{3.2}$$

The PD function calculates the average predicted outcome when the jth column of X is replaced with the value x_j. It is important to note that PD plots assume that the features in X_S are independent of the features in X_C. In general, PD plots provide a global model-agnostic technique that reveals the global relationship between a feature and the predicted output.

Marginal plots (M-plots) (Apley & Zhu, 2020) provide an alternative to PD plots by focusing on the conditional distribution, effectively addressing extrapolation issues that may arise from correlated features. The function of a marginal plot is defined as

$$\mathrm{M}_S = \mathbb{E}_{X_C|X_S}[\hat{f}(X_S, X_C)|X_S = x_S] = \int \hat{f}(X_S, X_C) d\mathbb{P}(X_C|X_S = x_S) \tag{3.3}$$

However, a limitation of M-plots arises when a feature of interest, j, is correlated with an unplotted feature k in X. In such cases, the M-plot for feature j show a mixed effect of both features j and k, even if feature j has no predictive power on the target variable. For this reason, M-plots have limited utility as tools for assessing main effects (Friedman, 2001; Apley & Zhu, 2020; Grömping, 2020).

3.2 Properties of PD Plots

Additive or Multiplicative Recovery Properties: If the dependence of $\hat{f}(X)$ on X_S is additive: $\hat{f}(X) = \hat{h}_S(X_S) + \hat{h}_C(X_C)$, then $\mathrm{PD}_S(X_S)$ is equal to $\hat{h}_S(X_S)$, up to an additive constant. If the dependence of $\hat{f}(X)$ on X_S is multiplicative: $\hat{f}(X) = \hat{h}_S(X_S) \cdot \hat{h}_C(X_C)$, then $\mathrm{PD}_S(X_S)$ is equal to $\hat{h}_S(X_S)$, up to an multiplicative constant factor (Friedman, 2001; Hastie et al., 2009). Marginal plots, in comparison, do not conform to these properties. Grömping (2020) argued that PD plots are more conceptually sound than M-plots or Accumulated Local Effect (ALE) plots – a popular alternative to PD plots.

Causal interpretations: It is possible to derive causal interpretations from black-box models using PD plots: PD plots estimate the causal effect of X_S on Y, provided that X_C satisfies the back-door criterion (Zhao & Hastie, 2021). Loftus et al. (2023) proposed causal dependence plots – a causal analogue of PD plot, based on the structural causal model, however, it can be a challenge to identify the structural causal model in real-world applications.

3.3 Limitations of PD Plots

Extrapolation: PD plots, along with other perturbation-based interpretation methods, does not account for the interaction effects among features. Consequently, when there is strong dependence among features, these methods may yield misleading interpretation results as they extrapolate to regions with little or no data (Hooker, 2004; Hooker et al., 2021). Furthermore, the selection of sampling points, such as the use of equidistant grids, can exacerbate the problem of extrapolation (Molnar et al., 2022; Krause et al., 2016).

Aggregation of Heterogeneous Effects: PD plots represent an average curve, but they can obscure opposite behaviors present within subsets of the data. For example, a flat PDP curve may suggest one of two scenarios: (1) the feature has no significant influence on the prediction, or (2) different subsets of the dataset exhibit opposing trends that offset each other when calculating the overall average (Angelini et al., 2023; Molnar et al., 2022).

Ignorance of Uncertainty: PD plot fails to consider the uncertainty in both its estimation and model fitting. For instance, its interpretations can be misleading in situations where models are underfitted or overfitted (Molnar et al., 2022). Instead, the uncertainty of PD plots can be estimated on bootstrap samples for a given model to construct confidence intervals (Cafri & Bailey, 2016).

Crucially, the first two limitations – the extrapolation behavior of correlated features and the aggregation of heterogeneous effects during the averaging process – are exploited in Section 4 to manipulate PD plot outputs.

4 The Process of Fooling Partial Dependence Plots

In this section, we outline our methodology for constructing an adversarial framework, denoted by $a(\mathbf{x})$, which is designed to replace the original ML model $f(\mathbf{x})$. Initially, we consider a simpler scenario of manipulating a single PD plot. Building on this foundation, we expand our methodology to simultaneously fool multiple PD plots within a unified framework.

4.1 Adversarial Framework a(x)

Our adversarial framework $a(\mathbf{x})$ takes the form:

$$a(\mathbf{x}) = (1 - c(\mathbf{x})) \cdot f(\mathbf{x}) + c(\mathbf{x}) \cdot g(\mathbf{x}). \tag{4.1}$$

The model $a(\mathbf{x})$ replaces the original machine learning model $f(\mathbf{x})$, which aims to maintain the prediction accuracy of the original model and generate manipulated PD plots.

Here, $c(\mathbf{x})$ is a classifier to distinguish between instances from the original feature space and those from the extrapolation domain. $g(\mathbf{x})$ is a function to provide specific outputs to form the desired PD plots. The framework predicts that an instance originates from the original feature space when

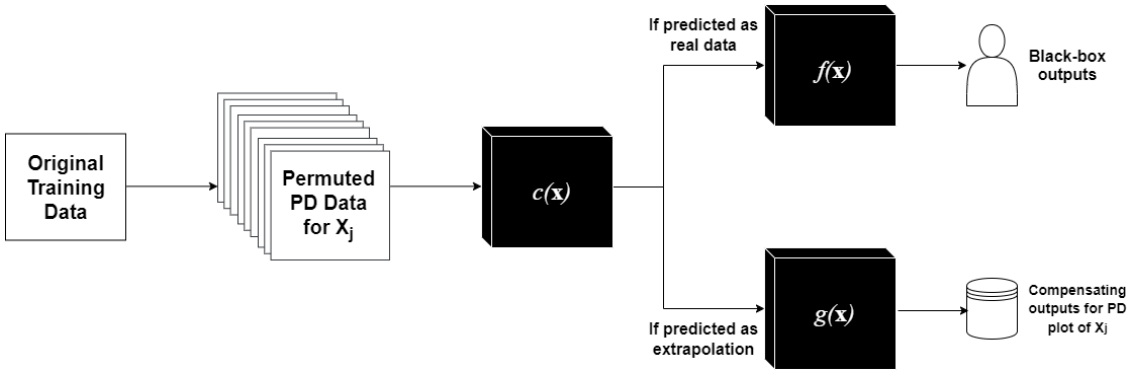

Figure 1: Adversarial Framework $a(\mathbf{x})$ for Manipulating the PD Plot of Feature X_j

$c(\mathbf{x}) = 0$ and outputs the black-box prediction $f(\mathbf{x})$ accordingly. Conversely, when $c(\mathbf{x}) = 1$, the instance is determined to come from the extrapolation domain, and $a(\mathbf{x})$ outputs $g(\mathbf{x})$ – the compensating outputs. Broadly, the framework uses $g(\mathbf{x})$ for instances in the extrapolation domain as identified by $c(\mathbf{x})$, aiming to compensate for the discriminatory performance of model $f(\mathbf{x})$ on real data when generating PD plot outputs. A simple scenario of this process is visually summarized in Figure 1.

The manipulated PD function for the j^{th} feature after adversarial attacks can be expressed as:

$$\widehat{\mathrm{PD}}_j^{\mathrm{adv}}(x_j) = \frac{1}{n}\sum_{i=1}^{n} \hat{a}(x_j, \mathbf{x}_{-j}^{(i)}) \tag{4.2}$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n} \hat{f}(x_j, \mathbf{x}_{-j}^{(i)}) \cdot (1 - \hat{c}(x_j, \mathbf{x}_{-j}^{(i)})) + \sum_{i=1}^{n} \hat{g}(x_j, \mathbf{x}_{-j}^{(i)}) \cdot \hat{c}(x_j, \mathbf{x}_{-j}^{(i)})\right) \tag{4.3}$$

We simplify this expression to:

$$\widehat{\mathrm{PD}}_j^{\mathrm{adv}}(x_j) = (1 - \hat{\lambda}_j(x_j))\hat{\rho}_j(x_j) + \hat{\lambda}_j(x_j)\hat{\gamma}_j(x_j), \tag{4.4}$$

where $\lambda_j(x_j)$ represents the proportion of permuted data identified as belonging to the extrapolation domain by $\hat{c}(\mathbf{x})$ for feature j at value x_j. The term $\rho_j(x_j)$ is the conditional PD function, which serves as an approximation of $\mathrm{M}_j(x_j)$ in (3.3), capturing the global relationship between the feature and the predicted output, considering non-extrapolation data only. $\gamma_j(x_j)$ denote the compensating output from model g for feature X_j at value x_j, when attempting to fool one PD plot at a time. We have $g(x_j, \mathbf{x}_{-j}^{(i)}) = \gamma_j(x_j)$. These components are estimated as follows:

$$\hat{\lambda}_j(x_j) = \frac{1}{n}\sum_{i=1}^{n} \hat{c}(x_j, \mathbf{x}_{-j}^{(i)}) \tag{4.5}$$

$$\hat{\rho}_j(x_j) = \frac{1}{n - n\hat{\lambda}_j(x_j)}\sum_{i=1}^{n} \hat{f}(x_j, \mathbf{x}_{-j}^{(i)}) \cdot (1 - \hat{c}(x_j, \mathbf{x}_{-j}^{(i)})) \tag{4.6}$$

Given these, if $\overline{\text{PD}}_j^{\text{adv}}(x_j)$ denotes the desired PD output set by modelers (attackers) for feature X_j at value x_j we derive:

$$g(x_j, \mathbf{x}_{-j}^{(i)}) = \hat{\gamma}_j(x_j) = \frac{\overline{\text{PD}}_j^{\text{adv}}(x_j) - (1 - \hat{\lambda}_j(x_j))\hat{\rho}_j(x_j)}{\hat{\lambda}_j(x_j)}. \tag{4.7}$$

In the following subsections, we discuss how to construct $c(\mathbf{x})$ and $g(\mathbf{x})$ and apply the adversarial framework to fool single and multiple PD plots.

4.2 Construct the Classifier $c(\mathbf{x})$

To build the adversarial framework, we commence with the construction of classifier $c(\mathbf{x})$ to distinguish between instances from the original feature space and those from the extrapolation domain. To construct such a classifier, we first generate an augmenting sample $\tilde{X} = \{\tilde{\mathbf{x}}^{(i)}\}_{i=1}^{\tilde{n}}$ following the approach proposed by Hooker & Rosset (2012), complementing the training of $c(\mathbf{x})$ with original data X. Let $\mathbb{P}(\tilde{\mathbf{x}})$ denote the joint (generative) feature distribution to generate $\tilde{X}$, and set it as a uniform distribution based on the empirical range of the training data:

$$\mathbb{P}(\tilde{\mathbf{x}}) = \mathbb{P}(\tilde{\mathbf{x}}; X) = \prod_{j=1}^{p} \mathcal{U}(\min_i(x_j^{(i)}) < \tilde{x}_j < \max_i(x_j^{(i)})), \tag{4.8}$$

where $\mathcal{U}(a, b)$ represents a uniform marginal distribution ranging from a to b, and $\min(x_j^{(i)})$ and $\max(x_j^{(i)})$ denote the minimum and maximum values of the jth feature in the training sample. In other words, each feature $\tilde{X}_j$ in $\tilde{X}$ is independently generated from the uniform marginal distribution of individual features.

The proposal of uniform $\mathbb{P}(\tilde{\mathbf{x}})$ was introduced by Hooker & Rosset (2012) for a different objective, aiming to provide regularization in regions of sparse data to control the model's extrapolation behavior and improve its prediction performance. In our study, the augmenting sample $\tilde{X}$ is drawn randomly and independently among features according to the uniform $\mathbb{P}(\tilde{\mathbf{x}})$. This process disrupts the dependence among features in the augmenting sample, leading to extrapolation. Subsequently, we combine the real data and augmenting (uniform) data, denoted as $X \cup \tilde{X}$, and assign labels $c(\mathbf{x}) = 0$ to instances in X and $c(\mathbf{x}) = 1$ to instances in $\tilde{X}$. The classifier $c(\mathbf{x})$ is trained using the combined dataset $X \cup \tilde{X}$ along with their corresponding labels to differentiate between real and uniform data. This approach enables $c(\mathbf{x})$ to learn the interrelationships within the real data X, thus effectively distinguishing instances that originate from the feature space defined by the training set from those in the extrapolation domain.

Our classifier $c(\mathbf{x})$ shares similarities with the out-of-distribution (OOD) classifier in Slack et al. (2020), which directly used the permuted data generated by LIME or kernel SHAP as their augmenting data. However, our approach diverges from directly using permuted data used in generating

PD plots to construct $c(\mathbf{x})$ for the following reasons: firstly, we do not treat all permuted PD data as instances of extrapolation[1]; secondly, our adversarial framework $a(\mathbf{x})$ uses only extrapolated permuted PD data as compensating outputs, rather than using all permuted PD data, thereby enhancing the efficiency and accuracy of the fooling process.

4.3 Apply the Adversarial Framework to Fool One PD Plot

Now we have constructed our adversarial framework in Equation 4.1 after constructing $c(\mathbf{x})$ and $g(\mathbf{x})$ in previous subsections. In this subsection, we introduce the process of implementing the adversarial attack to fool one PD plot at a time.

Let us denote the set of grid values which we will use to plot $\widehat{PD}_j(x_j)$ by $V_j = \{v_j^1, v_j^2, ..., v_j^{m_j}\}$, where m_j is the number of grid points and each v_j^p ($p \in \{1, 2, \ldots, m_j\}$) represents a specific grid value in V_j. For each $v_j^p \in V_j$, we create a permuted dataset by replacing the jth feature in all training data points with v_j^p. The set of all permuted datasets is then represented as:

$$\mathcal{P}_j = \left\{ \left(\mathbf{x}^{(i)}_{[j \to v_j^p]}, y^{(i)} \right) \right\}_{i=1,p=1}^{n,m_j}$$

Here, $\mathbf{x}^{(i)}_{[j \to v_j^p]}$ represents the feature vector of the i th instance with the jth feature replaced by v_j^p, done for all n instances and all m grid values. For the experimental results presented in Section 5, we carefully select grid values that are directly drawn from the training dataset, ensuring that each V_j is an observed value, thereby avoiding extrapolation within the range of the training data. This approach ensures that our algorithm's effectiveness is not artificially enhanced by extrapolated grid values. For further discussions and implementation details on this aspect, please refer to footnote 1 and Appendix C.2.

The process for manipulating a single PD plot is summarized in Algorithm 1. For each permuted observation where the jth feature is set to a specific value, the model $\hat{c}(\mathbf{x})$ determines if it originates from real data or extrapolation domain. If $\hat{c}(\mathbf{x})$ predicts the observation as real data, the output will remain as the black-box model's output $f(\mathbf{x})$; if it is predicted as extrapolation, the output switches to a predetermined value $\hat{\gamma}_j(x_j)$, based on the value of feature j. This procedure is formulated in (4.1) and illustrated in Figure 1. The same process applies when making predictions on the test data, and the goal is for our adversarial algorithm $a(\mathbf{x})$ to retain as many predictions of $f(\mathbf{x})$ as possible.

[1]We believe that the effective fooling of LIME (as compared to kernel SHAP) as observed in Slack et al. (2020) can be partially attributed to the fact that some permuted data used by LIME involve extrapolation within the training data range (e.g., if the feature 'age' consists only of integer values in all training data, while the permuted data include non-integer ages). This occurs because LIME is implemented by randomly perturbing the feature values of the original instance, which explains why directly using the permuted data works in Slack's approach. However, the fooling effectiveness of our algorithm does not benefit from this aspect (i.e., the term "permutation" does not refer to exactly the same thing in LIME and PD plots). For more details on the implementation of our algorithm, refer to Appendix C.

Algorithm 1 Fooling One Partial Dependence Plot

Input: Training data X, feature of interest X_j, grid values V_j, permuted PD data $\mathcal{P}_j$, target PD outputs $\overline{\text{PD}}_j^{\text{adv}}$, trained ML model $\hat{f}(\mathbf{x})$.

Output: Manipulated PD outputs $\widehat{\text{PD}}_j^{\text{adv}}(v_j^p)$ for each $v_j^p \in V_j$

Generate augmenting sample $\tilde{X}$
Label each $\mathbf{x} \in X \cup \tilde{X}$ with $l_c = 0$ if $\mathbf{x} \in X$, and $l_c = 1$ if $\mathbf{x} \in \tilde{X}$.
Train classifier $c(\mathbf{x})$ using $D_c = \{(\mathbf{x}, l_c) \mid \mathbf{x} \in X \cup \tilde{X}, l_c \in \{0, 1\}\}$.
for each $v_j^p \in V_j$ **do**
    Estimate $\hat{\lambda}_j(v_j^p)$ using $\mathcal{P}_j$ and $\hat{c}(\mathbf{x})$
    Estimate $\hat{\rho}_j(v_j^p)$ using $\mathcal{P}_j$, $\hat{c}(\mathbf{x})$ and $\hat{f}(\mathbf{x})$
    Estimate $\hat{\gamma}_j(v_j^p)$ using $\overline{\text{PD}}_j^{\text{adv}}(v_j^p)$, $\hat{\lambda}_j(v_j^p)$ and $\hat{\rho}_j(v_j^p)$
end for
for each $v_j^p \in V_j$ **do**
    Initialize an empty list A to store $\hat{a}(\mathbf{x})$ predictions at value v_j^p
    for each $\mathbf{x} \in \mathcal{P}_j$ where $x_j^{(i)} = v_j^p$ **do**
        $$\hat{a}(\mathbf{x}) = \begin{cases} \hat{f}(\mathbf{x}) & \text{if } \hat{c}(\mathbf{x}) = 0 \\ \hat{g}(\mathbf{x}) = \hat{\gamma}_j(v_j^p) & \text{if } \hat{c}(\mathbf{x}) = 1 \end{cases}$$
        Add $\hat{a}(\mathbf{x})$ to A
    end for
    Output $\widehat{\text{PD}}_j^{\text{adv}}(v_j^p)$ = average of A
end for

4.4 Construct $g(\mathbf{x})$ to Fool Multiple PD Plots

In practice, modelers may wish to fool multiple PD plots. However, independent manipulation of each PD plot could lead to inconsistent model predictions: if an observation is identified by $\hat{c}(\mathbf{x})$ as extrapolation, we need to select which feature's compensating outputs to use. To address this, we introduce a unified $g(\mathbf{x})$ model designed to ensure consistent prediction outputs while manipulating q PD plots simultaneously, where $1 < q \leq p$. Unlike the case of manipulating a single PD plot, we augment g with a specialized classifier $c_1(\mathbf{x})$, which is trained with $q+1$ class responses to allocate the appropriate compensating outputs to each PD plot accurately. The set of all potential classes for $c_1(\mathbf{x})$ is denoted as $G = \{G_1, G_2, \ldots, G_q, G_{\text{no}}\}$, including a base class G_{no} for real instances and classes G_1 to G_q corresponding to each of the q features targeted for manipulation. To train $c_1(\mathbf{x})$ we use the real data X along with the permuted PD data for the q targeted features identified as extrapolation by $\hat{c}(\mathbf{x})$, where target labels are given by which feature (or none) were permuted in producing each datum.

For clarity, we use k to denote the predicted index of the classifier $c_1(\mathbf{x})$ among the $q+1$ classes, distinct from index j for all p features in X, such that $\hat{c}_1(\mathbf{x}) = G_k, G_k \in G$. For a permuted observation $g(x_j, \mathbf{x}_{-j}^{(i)})$ classified by $c_1(\mathbf{x})$ as belonging to class G_k, the output of model g is then

given by

$$g(\mathbf{x}) = \begin{cases} \gamma_k(x_k) & \text{if } \hat{c}_1(\mathbf{x}) = G_k \\ f(\mathbf{x}) & \text{if } \hat{c}_1(\mathbf{x}) = G_{\text{no}}. \end{cases} \tag{4.9}$$

We thus divide the extrapolation part of feature space into regions associated with permuting each feature of interest. We anticipate that permuting different features leads to little overlap in the extrapolation region and can form adversarial function γ_k independently. Note that our framework has a two-step classifier: first applying $c(\mathbf{x})$ and then $c_1(\mathbf{x})$. This is to ensure that $f(\mathbf{x})$ remains unperturbed on the original data distribution – we will revert to the original model if either classifier tells us to. The process of fooling two plots is illustrated in Figure 2. Notably, in the scenario of fooling multiple PD plots, only $c(\mathbf{x})$ continues to affect the accuracy of $a(\mathbf{x})$ – the predictions retained of $a(\mathbf{x})$ from $f(\mathbf{x})$. The inclusion of $c_1(\mathbf{x})$ only affects the stability or uncertainty of $a(\mathbf{x})$ (how $\widehat{\text{PD}}_j^{\text{adv}}(x_j)$ deviates from $\overline{\text{PD}}_j^{\text{adv}}(x_j)$).

4.5 Apply the Adversarial Framework to Fool Multiple PD Plots

As summarized in Algorithm 1, let X_k represent an additional targeted feature for manipulation. We define $V_k = \{v_k^1, v_k^2, ..., v_k^{m_k}\}$ as the set of grid values corresponding to this feature. The process for manipulating two PD plots within a unified $g(\mathbf{x})$ model is outlined in Algorithm 2 and depicted in Figure 2, noting that our framework can be readily adapted for fooling multiple PD plots. In comparison, Algorithm 2 leverages compensating outputs derived from Algorithm 1, and the integration of $c_1(\mathbf{x})$ in Algorithm 2 introduces additional variability into the fooling process compared to Algorithm 1.

In summary, our framework employs extrapolated predictions identified by $c(\mathbf{x})$ to compensate for the discriminatory performance of model $f(\mathbf{x})$ on real data. It maintains the performance of $f(\mathbf{x})$ while concealing biases in the predictions on real data, rendering the framework seemingly neutral when interpreting the results through PD plots. Conceptually, we can regard the entire process as a unified black-box model, even though it involves the sequential construction of separate models. Instead of aggregating all model outputs as in (4.1), an alternative approach is to retrain $f(\mathbf{x})$ incorporating extrapolated permuted instances with their corresponding compensating outputs in the final step, but we note that this yields less fine control over the resulting precitions and partial dependence plots.

5 Experimental Results

5.1 Insurance Data

This study evaluates the performance of our algorithm on real-world insurance datasets. We used the `pg17trainpol` and `pg17trainclaim` datasets, which were obtained from the R package CAS-

Algorithm 2 Fooling Two Partial Dependence Plots Using a Single $g(\mathbf{x})$ Model

Input: Training data X, features of interest X_j, X_k, grid values V_j, V_k, permuted PD data $\mathcal{P}_j, \mathcal{P}_k$, target PD outputs $\overline{\text{PD}}_j^{\text{adv}}, \overline{\text{PD}}_k^{\text{adv}}$, trained ML model $\hat{f}(\mathbf{x})$.

Output: Manipulated PD outputs $\widehat{\text{PD}}_j^{\text{adv}}(v_j^p)$ for each $v_j^p \in V_j$, $\widehat{\text{PD}}_k^{\text{adv}}(v_k^p)$ for each $v_k^p \in V_k$

1: Perform steps 1-7 from Algorithm 1 for both features X_j and X_k to estimate $\hat{\lambda}_j, \hat{\rho}_j, \hat{\gamma}_j$ for each $v_j^p \in V_j$ and $\hat{\lambda}_k, \hat{\rho}_k, \hat{\gamma}_k$ for each $v_k^p \in V_k$.
2: Label each $\mathbf{x} \in X \vee (\mathbf{x} \in \mathcal{P}_j \cup \mathcal{P}_k \wedge \hat{c}(\mathbf{x}) = 1)$ with $l_g = 0$ if $x \in X$, $l_g = 1$ if $x \in \mathcal{P}_j \wedge \hat{c}(\mathbf{x}) = 1$, and $l_g = 2$ if $x \in \mathcal{P}_k \wedge \hat{c}(\mathbf{x}) = 1$.
3: Train classifier $c_1(\mathbf{x})$ using $D_g = \{(\mathbf{x}, l_g) \mid \mathbf{x} \in X \vee (\mathbf{x} \in \mathcal{P}_j \cup \mathcal{P}_k \wedge \hat{c}(\mathbf{x}) = 1), l_g \in \{0, 1, 2\}\}$.
4: **for** each $v_j^p \in V_j$ **do**
5: Initialize an empty list A to store $\hat{a}(\mathbf{x})$ predictions at value v_j^p
6: **for** each $\mathbf{x} \in \mathcal{P}_j$ where $x_j^{(i)} = v_j^p$ **do**
7:
$$\hat{a}(\mathbf{x}) = \begin{cases} \hat{f}(\mathbf{x}) & \text{if } \hat{c}(\mathbf{x}) = 0 \vee (\hat{c}(\mathbf{x}) = 1 \wedge \hat{c}_1(\mathbf{x}) = 0) \\ \hat{g}(\mathbf{x}) = \hat{\gamma}_j(v_j^p) & \text{if } \hat{c}(\mathbf{x}) = 1 \wedge \hat{c}_1(\mathbf{x}) = 1 \\ \hat{g}(\mathbf{x}) = \hat{\gamma}_k(x_k^{(i)}) & \text{if } \hat{c}(\mathbf{x}) = 1 \wedge \hat{c}_1(\mathbf{x}) = 2 \quad \text{(interpolate or extrapolate if needed)} \end{cases}$$
8: Add $\hat{a}(\mathbf{x})$ to A
9: **end for**
10: Output $\widehat{\text{PD}}_j^{\text{adv}}(v_j^p)$ = average of A
11: **end for**
12: Perform steps 4-11 to compute and output manipulated PD outputs $\widehat{\text{PD}}_k^{\text{adv}}(v_k^p)$ for each $v_k^p \in V_k$.

Note: When $\hat{c}_1(\mathbf{x})$ incorrectly identifies extrapolation to another feature, the compensating output $g(\mathbf{x})$ may require interpolation or extrapolation for continuous features not directly included in V_k (or V_j when fooling X_k).

datasets (Dutang & Charpentier, 2020) and used for the 2017 pricing game of the French institute of Actuaries. The data underwent preprocessing following the methodology proposed by Havrylenko & Heger (2022). Using this insurance dataset, we construct a model to predict claim frequency, denoted as $f(\mathbf{x})$. In this model, we use the number of claims (`claim_nb`) as the target variable. We also selected 14 features as explanatory variables. For details of the variables and summary statistics of the dataset, please refer to Appendix A.1.

Our study focuses on the driver's age (`drv_age1`) as the first focal variable for adversarial attack, which is highly correlated with other time or experience-related variables, including a strong correlation with the age of the driving license (`drv_age_lic1`). Younger drivers typically possess a shorter driving history and therefore cannot be experienced, while older drivers are less likely to have learned to drive at a late age and are thus unlikely to be amateurs, as shown in Figure 3.

In addition to exploring the driver's age, we also examine the vehicle value (`vh_value`) as a secondary focal variable for adversarial attack, which is known to be correlated with other essential vehicle characteristics. This choice allows us to demonstrate that our algorithm effectively manipulates PD plots of multiple features simultaneously using Algorithm 2 (see Appendix E.1 for results using Algorithm 1). Additionally, the target PD outputs are set as a flat line at the average predictions,

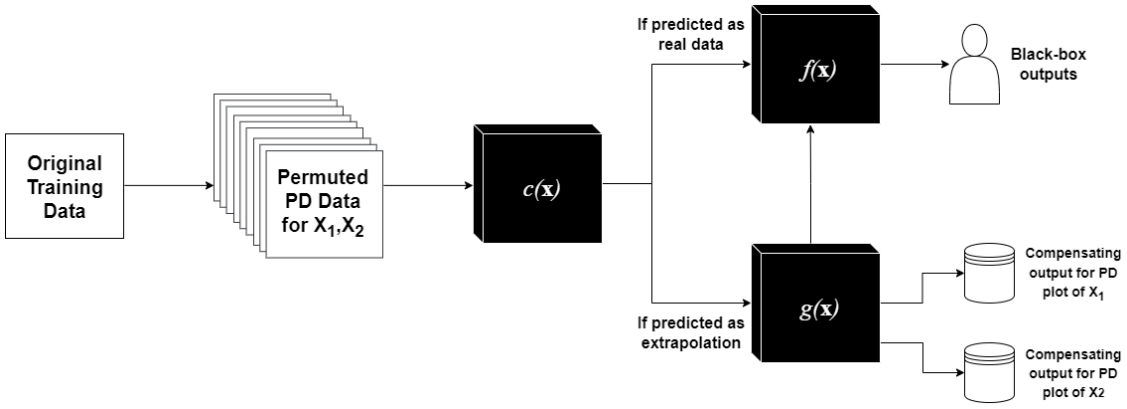

Figure 2: Adversarial Framework $a(\mathbf{x})$ for Manipulating the PD Plots of Features X_1 and X_2

$\overline{\text{PD}}_j^{\text{adv}}(v_j^p) = \hat{f}_{\text{avg}}(\mathbf{x})$ for each $v_j^p \in V_j$. The choice of a flat target line is illustrative and can be easily extended to various targeted manipulations. For instance, we could perform a surgical-like manipulation on the PD plot for age, specifically targeting older ages, and only slightly adjust its PD values.

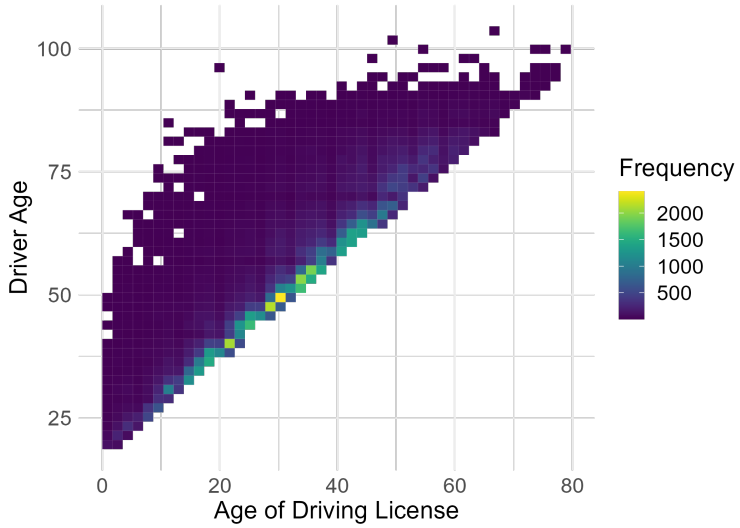

Figure 3: Relationship Between Driver's Age and Age of Driving License in the Insurance Dataset

The classifier $c(\mathbf{x})$ was constructed using the Keras package in R with a neural network architecture that comprises four fully connected layers with 40, 20, 10, and 1 units, respectively, and uses the rectified linear unit (ReLU) activation function. Likewise, the classifier $g(\mathbf{x})$ was implemented with a similar neural network architecture but was trained based on permuted PD data. During the training of our neural networks, we employed techniques such as dropout and batch normalization to mitigate overfitting and enhance overall performance. The specifics of all neural network models trained in Section 5 are detailed in Appendix D.2. The black-box insurance model $f(\mathbf{x})$ employed extreme gradient boosting (XGBoost) for claim frequency prediction. To optimize the performance

of XGBoost, hyperparameters were carefully selected through a grid search approach with details provided in Appendix D.1. Figures 4 and 5 showcase the resulting PD plots of the driver's age and the vehicle's value. It is important to note that the grid values for `vh_value` were chosen based on the quantiles of the feature's distribution, rather than using equally spaced points. Notably, our adversarial attacks have proven to be effective, as is visually evident from the observed results.

For this evaluation, we divided the `pg17trainpol` and `pg17trainclaim` datasets into five folds. In each iteration, we assigned four folds of the data for training the adversarial framework and reserved the remaining one fold for testing purposes. The resulting PD plots of the driver's age and the vehicle's value calculated using the testing set are displayed in Figures 4 and 5, respectively, under the condition that fold 1 is held out, and the threshold of $c(\mathbf{x})$ is set to 0.5. Our adversarial attacks have exhibited remarkable effectiveness, clearly evident from the visual analysis of the obtained results.

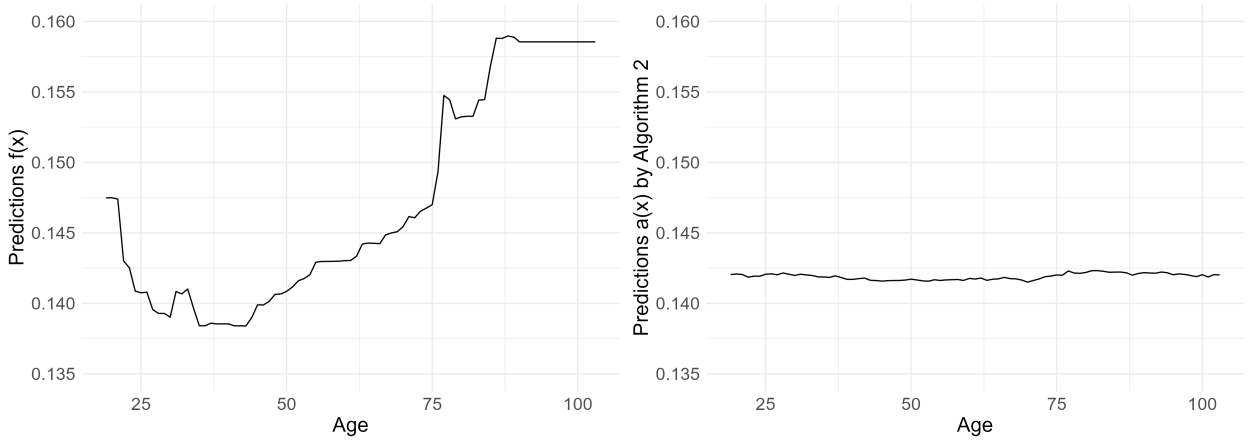

Figure 4: PD Plots for Age before Attack (Left) and after Attack (Right), Insurance Data

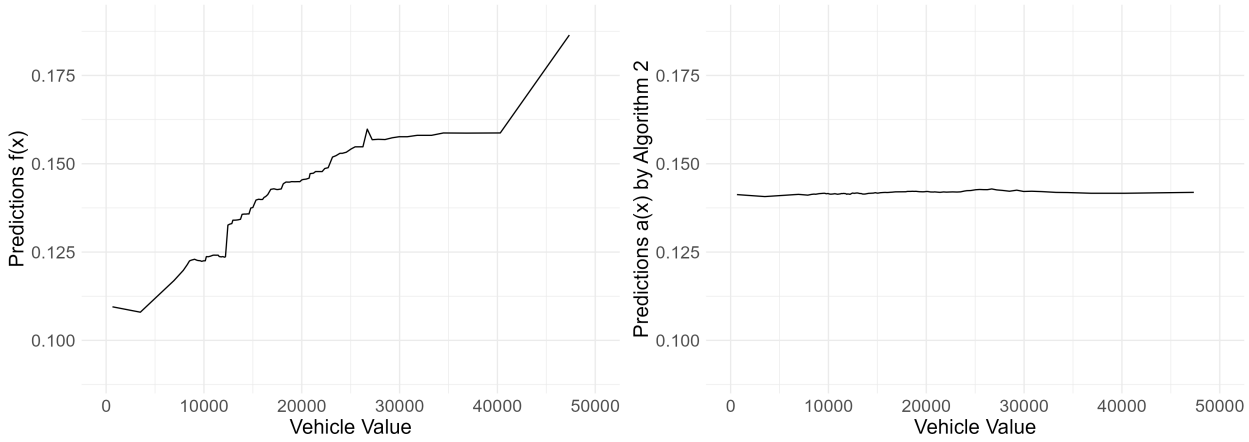

Figure 5: PD Plots for Vehicle Value before Attack (Left) and after Attack (Right), Insurance Data

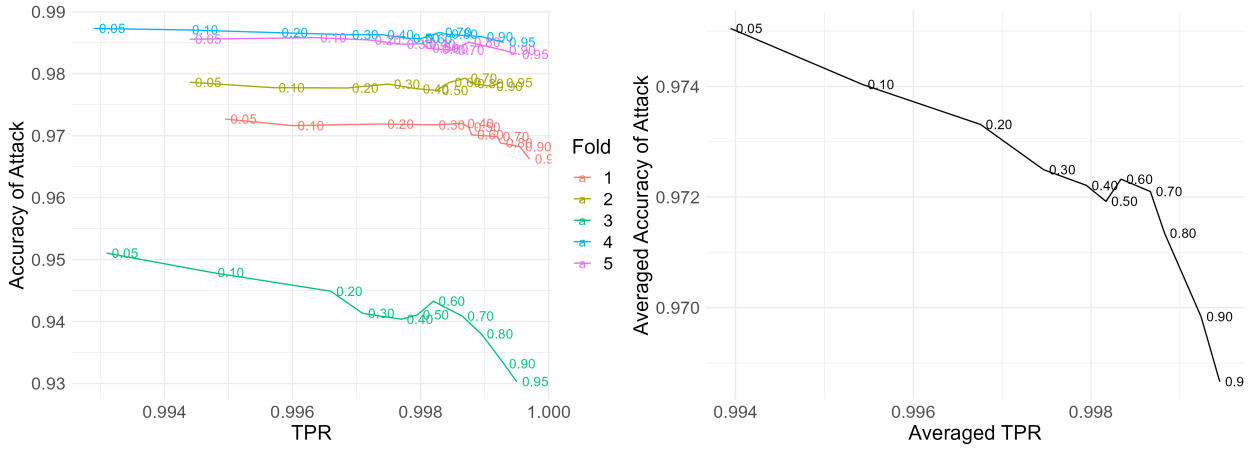

Figure 6: TPR versus Accuracy of Attack across Thresholds $\{0.05, 0.1, 0.2, \ldots, 0.8, 0.9, 0.95\}$ (labeled next to the lines) for PD Plots of Age, Insurance Data

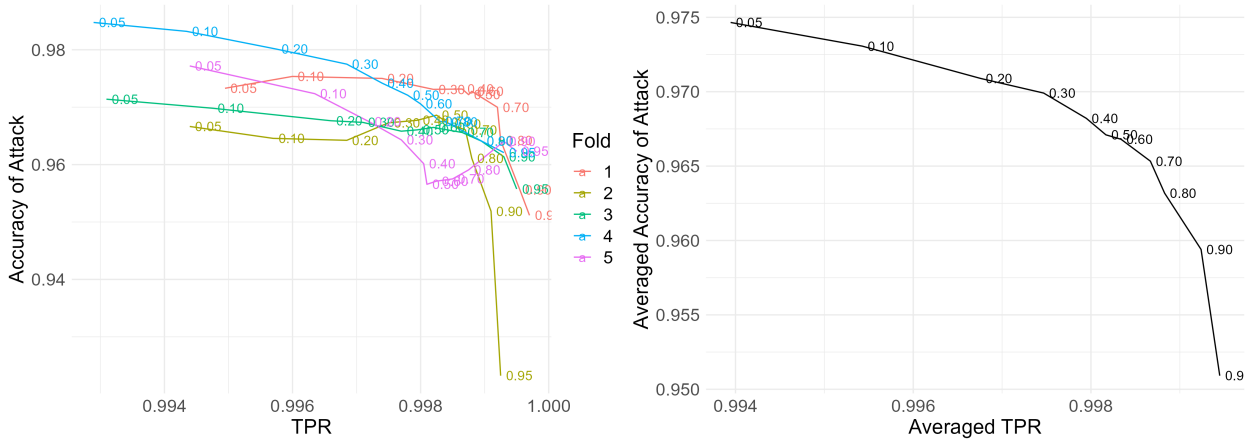

Figure 7: TPR versus Accuracy of Attack across Thresholds $\{0.05, 0.1, 0.2, \ldots, 0.8, 0.9, 0.95\}$ (labeled next to the lines) for PD Plots of Vehicle Value, Insurance Data

Effect of Different Thresholds for c(x)

In our adversarial framework, there exists an inevitable trade-off between the combined true positive rate of $c(\mathbf{x})$ and $c_1(\mathbf{x})$, which measures how closely $a(\mathbf{x})$ retain the performance of $f(\mathbf{x})$, and the accuracy of attack, measuring the effectiveness of adversarial attacks on PD plots. In our evaluation, we assess the impact of this trade-off on the hold-out test set by varying different thresholds for $\mathrm{c}(\mathbf{x})$. A lower threshold results in more data being labeled as extrapolation by $c(\mathbf{x})$, enhancing the efficiency and accuracy of the fooling process but at the cost of reducing the proportion of $f(\mathbf{x})$ retained by $a(\mathbf{x})$. We define metrics for both true positive rate (TPR) and accuracy of attack as

follows:

$$\text{True Positive Rate (TPR)} = \frac{1}{|X_{\text{test}}|} \sum_{\mathbf{x} \in X_{\text{test}}} \left(\mathbb{I}(\hat{c}(\mathbf{x}) = 0) + \mathbb{I}(\hat{c}(\mathbf{x}) = 1 \wedge \hat{c}_1(\mathbf{x}) = G_{\text{no}}) \right) \tag{5.1}$$

$$\text{Accuracy of Attack} = 1 - \frac{\|\widehat{\text{PD}}_j^{\text{adv}}(v_j^p) - \overline{\text{PD}}_j^{\text{adv}}(v_j^p)\|}{\|\widehat{\text{PD}}_j^{\text{orig}}(v_j^p) - \overline{\text{PD}}_j^{\text{adv}}(v_j^p)\|} \tag{5.2}$$

Here, X_{test} refers to the hold-out test set. The true positive rate is defined as the proportion of data identified as non-extrapolation by either $c(\mathbf{x})$ or $c_1(\mathbf{x})$, indicating that the proportion of $f(\mathbf{x})$ outputs are accurately preserved in $a(\mathbf{x})$. The accuracy of the attack measures the average difference between the PD values estimated by the adversarial model and the desired PD values, scaled by the difference between the original PD values and the desired PD values. Higher values of these measures indicate higher proportion of f($\mathbf{x}$) retained in a($\mathbf{x}$) or more accurate fooling outcomes. The results, based on the tracks of five folds, are presented in Figures 6 and 7. Notably, a generally decreasing trend is observed, highlighting the trade-off between TPR and accuracy of attack for both the PD plots of age and vehicle value. We attribute the increasing pattern, as seen in Figure 6 when thresholds shift from 0.5 to 0.7, to the fact that age is less significant in modeling, resulting in less accurate manipulated PD plots.

5.2 COMPAS Data

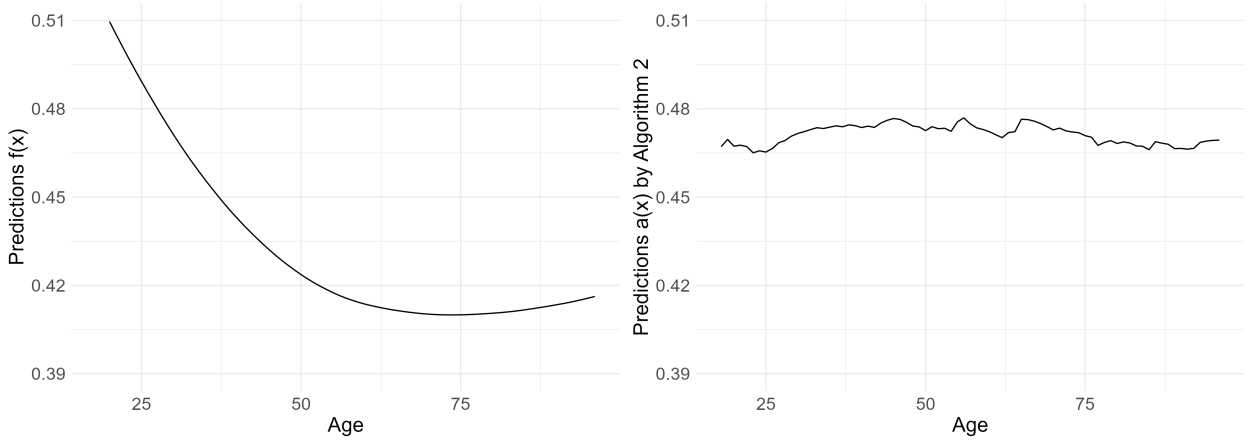

Figure 8: PD Plots for Age before Attack (Left) and after Attack (Right), COMPAS Data

We further evaluate our framework using the COMPAS dataset (Angwin et al., 2016), a popular fairness dataset in fair machine learning literature. This dataset comprises criminal offenders screened in Florida, U.S., during 2013–2014. The response variable, `two_year_recid`, indicates whether a person recidivated within two years after the screening, while sex and race are considered sensitive attributes and the remaining variables serve as predictors. We use the COMPAS data from the fairML R package, which follows the preprocessing steps outlined in Komiyama et al. (2018). Variables, summary statistics, and modeling details for the COMPAS dataset used are

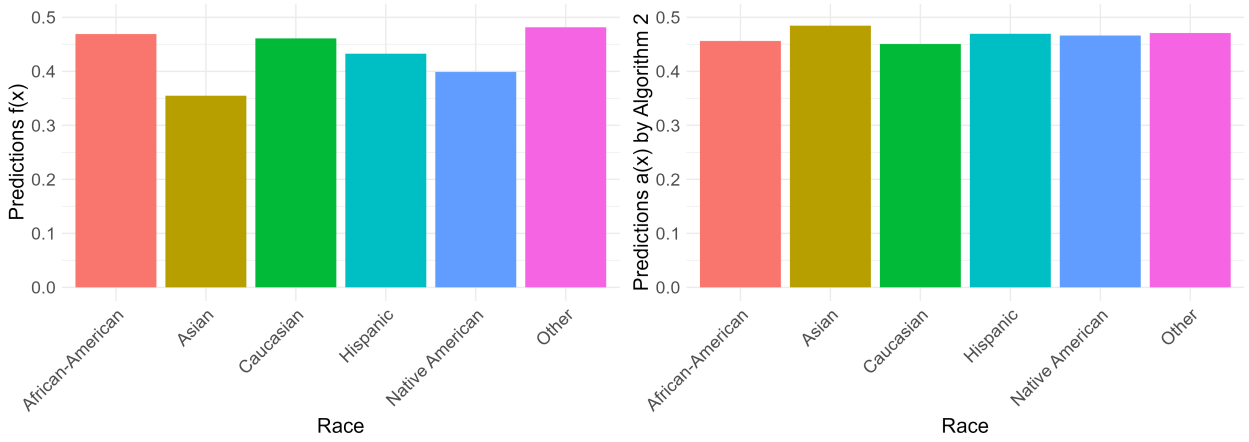

Figure 9: PD Plots for Race before Attack (Left) and after Attack (Right), COMPAS Data

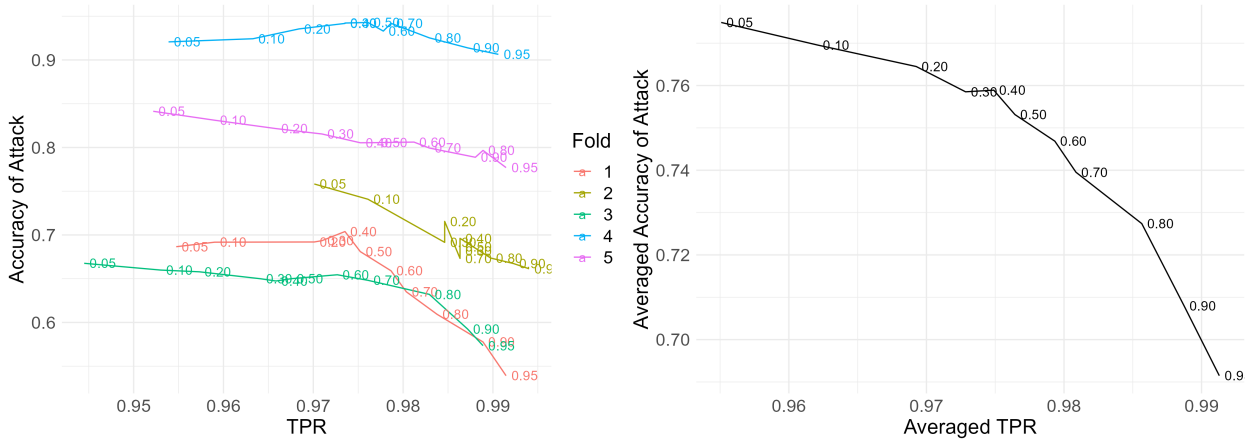

Figure 10: TPR versus Accuracy of Attack across Thresholds $\{0.05, 0.1, 0.2, \ldots, 0.8, 0.9, 0.95\}$ (labeled next to the lines) for PD Plots of Age, COMPAS Data

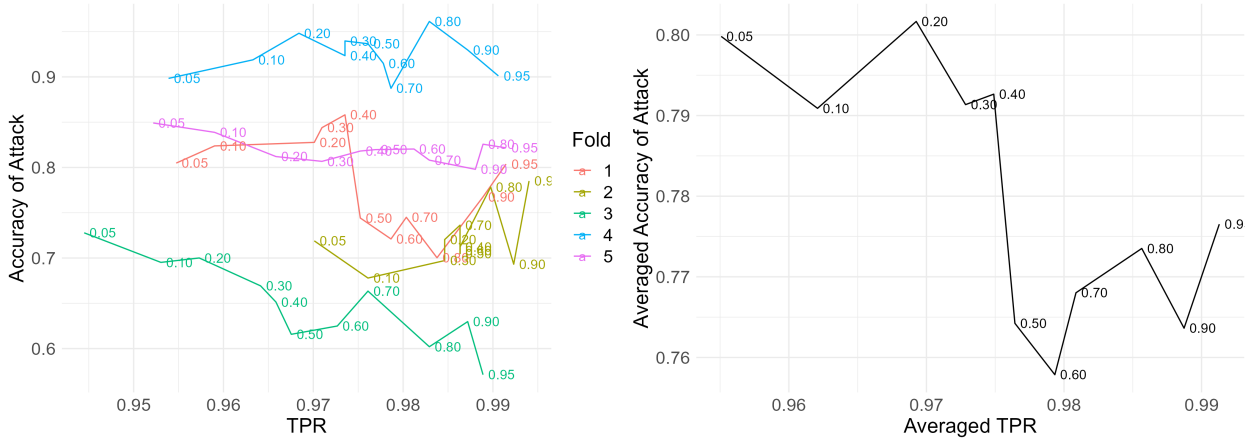

Figure 11: TPR versus Accuracy of Attack across Thresholds $\{0.05, 0.1, 0.2, \ldots, 0.8, 0.9, 0.95\}$ (labeled next to the lines) for PD Plots of Race, COMPAS Data

presented in Appendix A.2 and Appendix D.2.

In our modeling of $f(\mathbf{x})$ and $a(\mathbf{x})$, we intentionally include sex and race to present an example of manipulating a categorical feature. Our study primarily focuses on age and race as our focal variables, and we use Algorithm 2 to manipulate their PD plots within a unified framework. Similar to the insurance data, we set the target PD outputs as a flat line. Figures 8 and 9 illustrate that older individuals and those of Asian and Native American descent exhibit a lower probability of reoffending. We construct the classifiers $c(\mathbf{x})$ and $g(\mathbf{x})$ using a process similar to that employed for the insurance data. In the case of the COMPAS data, $f(\mathbf{x})$ is trained using a neural network architecture similar to that of $c(\mathbf{x})$ and $g(\mathbf{x})$. Remarkably, our model maintains strong performance as demonstrated in Figures 10 and 11, even though the sample size of the COMPAS data (5,836 versus 100,000) is smaller compared to the insurance data and the highest Pearson correlation between the focal variable age and the remaining variables is much smaller (-0.579891 versus 0.920605). For reference, an examination of the correlation matrix for all features used in the auto insurance claims and COMPAS datasets is presented in Appendix B.

5.3 A Simulated Example

As the fooling procedure depends on the interrelationships among features, it is necessary to explore whether only high correlations can facilitate effective fooling. We present a simulated example to demonstrate that an accumulation of weak correlations among features is sufficient for the attacks. The response variable is generated by

$$y^{(i)} = x_1^{(i)} + x_2^{(i)} + x_3^{(i)} + x_4^{(i)} + x_5^{(i)} + 0 \cdot x_6^{(i)} + \epsilon^{(i)} \tag{5.3}$$

where $\epsilon^{(i)} \sim N(0, 0.5^2)$ adds noise to the process and x_6 is set to have no influence on y. The features $\{X_j\}_{j=1}^6$ are normally distributed with mean 0 and standard deviation 1, and a correlation of 0.3 between each pair. Neural networks were used to construct $c(\mathbf{x})$ and $f(\mathbf{x})$, but this time, the threshold of $c(\mathbf{x})$ is also optimized and fixed at 0.955 after assigning different weights to the two classes during the training of $c(\mathbf{x})$, as the training data of $c(\mathbf{x})$ is highly imbalanced. Predictions of $c(\mathbf{x})$ for correctly classifying real data hold more significance than on uniform data. Figure 12 shows the scatter plot between the generated X_1 and X_2, where the correlation of 0.3 is hardly distinguishable by the eye.

We set the target PD outputs as a flat line for X_1 and an increasing line with a slope of 2 for X_6. The entire process is performed five times on five-fold validation data. The average TPR retained across five folds is 92.09%, and the PD plots of features X_1 and X_6 before and after the attacks are displayed in Figures 13 and 14. The observed curvature in the manipulated PD plots can be attributed to the relatively high misclassification errors of $c_1(\mathbf{x})$, indicating the challenge in identifying extrapolations for feature values near their mean. Our adversarial framework maintains effectiveness, despite the presence of only weakly correlated features in the data.

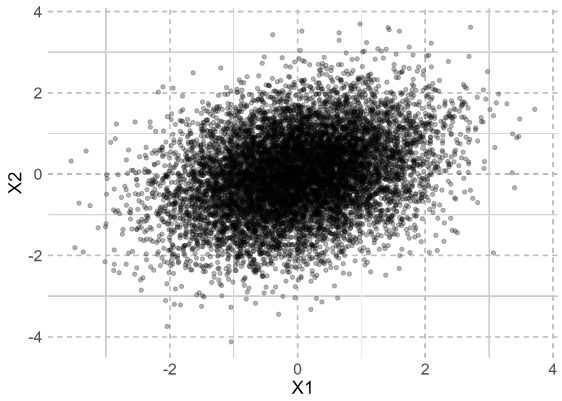

Figure 12: Scatter Plot of Simulated X_1 and X_2

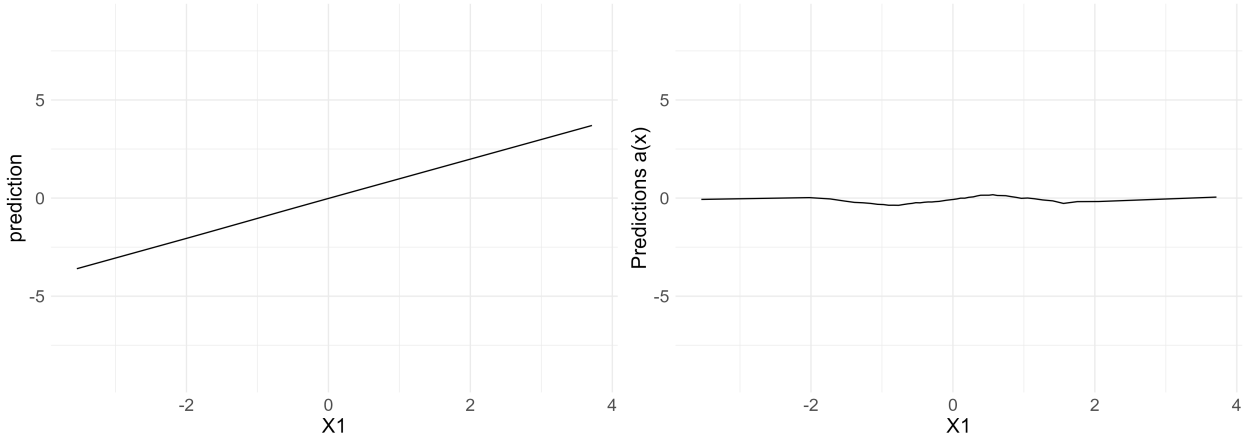

Figure 13: PD Plots for X_1 before Attack (Left) and after Attack (Right), Simulated Data

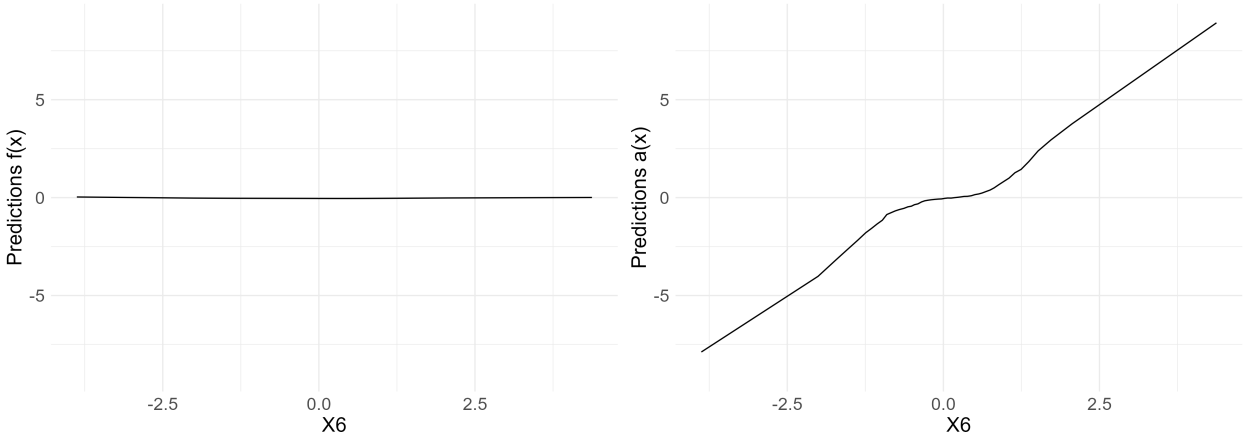

Figure 14: PD Plots for X_6 before Attack (Left) and after Attack (Right), Simulated Data

In Figure 15, we explore the permutation feature importance (PFI) of our adversarial algorithm $a(x)$ as a by-product of our adversarial attacks on PD plots. Before the attack, features X_1 to X_5 demonstrate roughly equal importance in $f(x)$ as we expected. After the attacks targeting

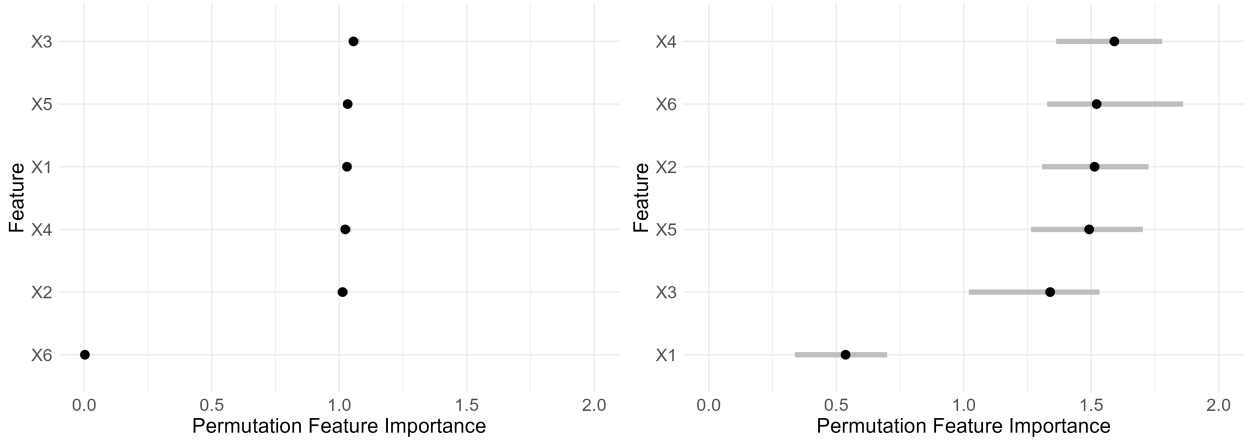

Figure 15: Permutation Feature Importance for $f(\mathbf{x})$ before Attack (Left) and $a(\mathbf{x})$ after Attack (Right), Simulated Data. The grey lines represent the range from the 10th to the 90th percentile of PFI estimates across fifty runs, with median values indicated by black dots.

X_1 and X_6, the importance of X_6 significantly increases, whereas X_1 shows a reduced importance relative to the other features. Additionally, the PFI estimates show increased variability post-attack, attributed to the use of extreme compensating outputs in our fooling algorithm. A similar analysis of PFI for the insurance and COMPAS datasets is presented in Appendix E.2.

6 Insights for Regulators and Practitioners

When to use interpretation tools for black-box models? There are three primary approaches for practitioners to develop interpretable AI models (EIOPA, 2021): 1) Use traditional interpretable models like Generalized Linear Models (GLMs) or Generalized Additive Models (GAMs); 2) Adopt a hybrid approach by leveraging black-box models solely for feature engineering; 3) Implement black-box models while supplementing them with interpretable tools. We argue that resorting to the third strategy—relying on interpretation methods to explain black-box models—should only be considered if accuracy is critical and the interpretation requirement is relatively low in the application context. Not just PD plots, but all interpretation tools have pitfalls. Practitioners should avoid the overuse of black-box models if interpretable models can achieve the same level of model performance (Rudin, 2019). In certain contexts, alternatives like GAMs can provide similar predictive accuracy as complex black-box models while offering greater interpretability (Lou et al., 2013; Caruana et al., 2015).

How to mitigate adversarial attacks on PD plots? First, enhance PD plots with the addition of Individual Conditional Expectation (ICE) curves (Grömping, 2020; Molnar et al., 2022), which are first proposed by Goldstein et al. (2015). Mathematically, an ICE curve for a feature j for the i^{th} observation is defined as:

$$\text{ICE}_j^{(i)}(x_j) = \hat{f}(x_j, \mathbf{x}_{-j}^{(i)}) \tag{6.1}$$

Essentially, a PD curve in (3.2) is the average of n ICE curves in (6.1) (see Appendix F for further discussion on ICE plots). As illustrated in Figure 16, integrating ICE curves with the PD curve for age helps identify anomalies post-attack, although this is effective only when all ICE plots are shown. It is important to note that the 10th and 90th percentile curves do not reveal the attack to nearly the same extent. Additionally, it may also be possible to vary our naive attack to make these plots less visually apparent.

Second, prior to applying interpretation methods, carefully assess the dependencies between features in the data (Molnar et al., 2022). For example, NAIC (2022) suggests that regulators obtain a correlation matrix for all predictor variables (see also SOA (2021)).

Third, to address interpretability concerns and maintain transparency, practitioners may consider using traditional interpretable models like GLMs or GAMs if the benefits of adopting black-box models are not substantial (Rudin, 2019). GAMs can also be used to approximate the $f(\mathbf{x})$ in a manner similar to (4.2) and discrepancies between GAMs and their corresponding PDPs may provide reasons to suspect an attack.

Fourth, practitioners may adopt a hybrid approach that leverages black-box models solely for feature engineering (EIOPA, 2021).

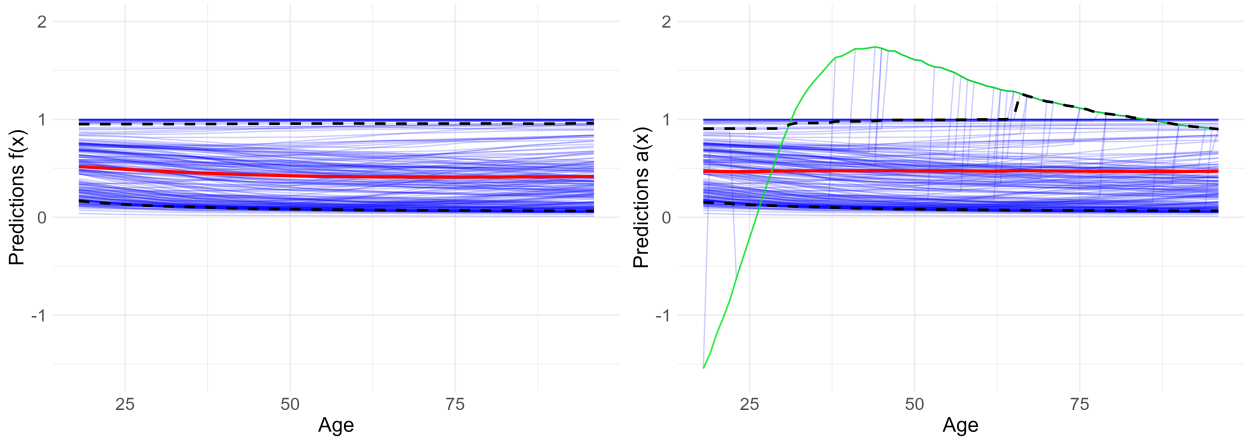

Figure 16: PD Plots (in Red, same as in Figure 8) for Age with ICE Curves (in Blue) Before (Left) and After (Right) Attack, Including 10th and 90th Percentiles of ICE Curves (in Black) and Compensating Outputs $\hat{\gamma}_{age}$ (in Green), COMPAS Data

How to use PD plots responsibly in practice? We advise against employing PD plots as a means to validate the fairness or non-discrimination of sensitive attributes. This is particularly important in adversarial scenarios as characterized by Bordt et al. (2022), where the stakeholders providing and utilizing interpretation methods have opposing interests and incentives. For instance, while age may be legitimately used, it remains a sensitive variable in black-box modeling, with persistent concerns over its impact on distinct age groups. Our empirical evidence suggests that discriminatory tendencies based on age, especially towards the elderly or youth, can be intentionally hidden, making the model appear neutral through tools like PD plots while preserving

nearly all predictions of the original black-box models. Additionally, practitioners should be aware of PD plots' limitations in interpreting black-box models. We advocate for the use of multiple interpretation tools to achieve a more holistic understanding of the model's feature effects. For a detailed exploration of alternatives to PD plots, we refer readers to Appendix F.

7 Conclusions

Potential Impacts of the Work: Our framework can be extended to fool permutation-based feature importance. By constructing $g(\mathbf{x})$ to reply on some legitimate factors, we can conceal the significant impact of a sensitive variable X_j previously in $f(\mathbf{x})$. However, it is difficult to pre-define the target of the desired feature importance value for X_j as we did for PD plots. Permutation-based feature importance can be employed in regulatory fairness checks to assess whether the model assigns significant importance to controversial variables, such as race or sex, in making predictions (a concern analogous to the one presented by Heo et al. (2019)). A real regulatory example of feature importance can be found in California's private passenger automobile insurance regulation (California Code of Regulations, Title 10, Section 2632.5), which stipulates that three mandatory rating variables must carry greater weight in pricing compared to other optional rating variables. Our adversarial framework can potentially hide the significant importance of optional rating variables used and satisfy the regulatory requirements.

Limitations of the Work: First, Our adversarial framework relies on the assumption that the target feature is not independent of other features in the dataset. Although this assumption might not universally hold, it is often plausible in data-rich contexts. As illustrated in our simulated example, our framework remains efficient even with weak feature correlations among features. Second, the success of the attacks is contingent on the level of information available to both attackers and defenders. We make the assumption that attackers (practitioners) have access to all relevant information, while defenders (regulators) have limited information, such as not being able to inspect the underlying codes. Third, there exists a possibility that the compensatory output, denoted by $\gamma_j(x)$, could potentially fall outside the expected domain or range. Fourth, when attempting to fool multiple PD plots, the compensatory output for each feature is set independently before constructing the classifier $c_1(\mathbf{x})$. Though it is feasible to calculate these estimates post-training of $c_1(\mathbf{x})$ to incorporate its uncertainty estimation, this approach is notably more complex.

References

Angelini, M., Blasilli, G., Lenti, S., & Santucci, G. (2023). A visual analytics conceptual framework for explorable and steerable partial dependence analysis. *IEEE Transactions on Visualization and Computer Graphics*.

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine bias.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *82*(4), 1059–1086.

Baniecki, H., & Biecek, P. (2022). Manipulating shap via adversarial data perturbations (student abstract). In *Proceedings of the aaai conference on artificial intelligence* (Vol. 36, pp. 12907–12908).

Baniecki, H., & Biecek, P. (2023). Adversarial attacks and defenses in explainable artificial intelligence: A survey. *arXiv preprint arXiv:2306.06123*.

Baniecki, H., Kretowicz, W., & Biecek, P. (2023). Fooling partial dependence via data poisoning. In *Machine learning and knowledge discovery in databases: European conference, ecml pkdd 2022, grenoble, france, september 19–23, 2022, proceedings, part iii* (pp. 121–136).

Berk, R. A., & Bleich, J. (2013). Statistical procedures for forecasting criminal behavior: A comparative assessment. *Criminology & Pub. Pol'y*, *12*, 513.

Bordt, S., Finck, M., Raidl, E., & von Luxburg, U. (2022). Post-hoc explanations fail to achieve their purpose in adversarial contexts. In *Proceedings of the 2022 acm conference on fairness, accountability, and transparency* (pp. 891–905).

Bücker, M., Szepannek, G., Gosiewska, A., & Biecek, P. (2022). Transparency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society*, *73*(1), 70–90.

Cafri, G., & Bailey, B. A. (2016). Understanding variable effects from black box prediction: Quantifying effects in tree ensembles using partial dependence. *Journal of Data Science*, *14*(1), 67–95.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1721–1730).

Delcaillau, D., Ly, A., Papp, A., & Vermet, F. (2022). Model transparency and interpretability: survey and application to the insurance industry. *European Actuarial Journal*, 1–42.

Dimanov, B., Bhatt, U., Jamnik, M., & Weller, A. (2020). You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., & Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, *32*.

Dutang, C., & Charpentier, A. (2020). Package 'casdatasets'. *Christophe Dutang and Arthur Charpentier*.

EIOPA. (2021). Artificial intelligence governance principle: Towards ethical and trustworthyartificial intelligence in the european insurance sector.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Ghorbani, A., Abid, A., & Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 3681–3688).

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, *24*(1), 44–65.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grömping, U. (2020). Model-agnostic effects plots for interpreting machine learning models. *Reports in Mathematics, Physics and Chemistry, Department II, Beuth University of Applied Sciences Berlin Report*, *1*, 2020.

Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Systems with Applications*, *39*(3), 3659–3667.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Havrylenko, Y., & Heger, J. (2022). Detection of interacting variables for generalized linear models via neural networks. *arXiv preprint arXiv:2209.08030*.

Henckaerts, R., Côté, M.-P., Antonio, K., & Verbelen, R. (2021). Boosting insights in insurance tariff plans with tree-based machine learning methods. *North American Actuarial Journal*, *25*(2), 255–285.

Heo, J., Joo, S., & Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, *32*.

Hooker, G. (2004). Diagnosing extrapolation: Tree-based density estimation. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 569–574).

Hooker, G., Mentch, L., & Zhou, S. (2021). Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing*, *31*, 1–16.

Hooker, G., & Rosset, S. (2012). Prediction-based regularization using data augmented regression. *Statistics and Computing*, *22*(1), 237–249.

Komiyama, J., Takeda, A., Honda, J., & Shimao, H. (2018). Nonconvex optimization for regression with fairness constraints. In *International conference on machine learning* (pp. 2737–2746).

Krause, J., Perer, A., & Ng, K. (2016). Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 5686–5697).

Kuo, K., & Lupton, D. (2020). Towards explainability of machine learning models in insurance pricing. *arXiv preprint arXiv:2003.10674*.

Laberge, G., Aïvodji, U., & Hara, S. (2022). Fooling shap with stealthily biased sampling. *arXiv preprint arXiv:2205.15419*.

Lee, S. C., & Lin, S. (2018). Delta boosting machine with application to general insurance. *North American Actuarial Journal*, *22*(3), 405–425.

Lemmens, A., & Croux, C. (2006). Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, *43*(2), 276–286.

Loftus, J. R., Bynum, L. E., & Hansen, S. (2023). Causal dependence plots for interpretable machine learning. *arXiv preprint arXiv:2303.04209*.

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (pp. 623–631).

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*.

Matuszelański, K., & Kopczewska, K. (2022). Customer churn in retail e-commerce business: Spatial and machine learning approach. *Journal of Theoretical and Applied Electronic Commerce Research*, *17*(1), 165–198.

Mohanty, A., & Mishra, S. (2022). A comprehensive study of explainable artificial intelligence in healthcare. In *Augmented intelligence in healthcare: A pragmatic and integrated analysis* (pp. 475–502). Springer.

Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C. A., ... Bischl, B. (2022). General pitfalls of model-agnostic interpretation methods for machine learning models. In *xxai-beyond explainable ai: International workshop, held in conjunction with icml 2020, july 18, 2020, vienna, austria, revised and extended papers* (pp. 39–68).

NAIC. (2020). *Casualty actuarial and statistical (c) task force - regulatory review of predictive models white paper.* `https://content.naic.org/sites/default/files/CA-WP_1.pdf`.

NAIC. (2022). *Appendix b-trees – information elements and guidance for a regulator to meet best practices' objectives (when reviewing tree-based models).* https://content.naic.org/sites/default/files/inline-files/CASTF%20Tree-based%20Model%20Appendix%20%28B-Trees%29.pdf.

Panigutti, C., Hamon, R., Hupont, I., Fernandez Llorca, D., Fano Yela, D., Junklewitz, H., ... others (2023). The role of explainable ai in the context of the ai act. In *Proceedings of the 2023 acm conference on fairness, accountability, and transparency* (pp. 1139–1150).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1135–1144).

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

Slack, D., Hilgard, A., Lakkaraju, H., & Singh, S. (2021). Counterfactual explanations can be manipulated. *Advances in neural information processing systems*, *34*, 62–75.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the aaai/acm conference on ai, ethics, and society* (pp. 180–186).

SOA. (2021). *Interpretable machine learning for insurance.* https://www.soa.org/globalassets/assets/files/resources/research-report/2021/interpretable-machine-learning.pdf.

Stanford HAI. (2019). *Artificial intelligence index report 2019.* https://hai.stanford.edu/sites/default/files/ai_index_2019_report.pdf.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Szepannek, G., & Lübke, K. (2023). How much do we see? on the explainability of partial dependence plots for credit risk scoring.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, *7*(2), 76–99.

Xie, S. (2021). Improving explainability of major risk factors in artificial neural networks for auto insurance rate regulation. *Risks*, *9*(7), 126.

Yang, Y., Qian, W., & Zou, H. (2018). Insurance premium prediction via gradient tree-boosted tweedie compound poisson models. *Journal of Business & Economic Statistics*, *36*(3), 456–470.

Zhao, Q., & Hastie, T. (2021). Causal interpretations of black-box models. *Journal of Business & Economic Statistics*, *39*(1), 272–281.

Supplementary Materials: Why You Should Not Trust Interpretations in Machine Learning: Adversarial Attacks on Partial Dependence Plot

A Datasets Used

A.1 Data Preparation for Insurance Data

As mentioned in the main text, the data preprocessing was conducted in accordance with the methodology proposed by Havrylenko & Heger (2022). The steps include:

1. For the claims dataset `pg17trainclaim`, calculate the sum of claim numbers grouped by `id_client`, `id_vehicle`, and `id_year` for each row.
2. Merge the `pg17trainclaim` dataset with the `pg17trainpol` dataset using `id_client`, `id_vehicle`, and `id_year` as keys.
3. Replace NA values in `claim_nb` with zero.
4. Filter out observations where the age of the driving license (`drv_age_lic1`) is less than the driver's age (`drv_age1`) minus 17.
5. Remove observations with NA values in `vh_age`.
6. Impute mean values for zero entries in `vh_value` and `vh_weight`, based on `vh_make` and `vh_model` categories.
7. Reduce the 101 categories of `vh_make` to 18 by merging those with similar average responses, applying k-means clustering for the aggregation. Label this newly aggregated feature as `vh_makenew`.

Following these preprocessing steps, the merged dataset comprises 99,918 observations. Due to the absence of exposure information in the dataset, we introduce the feature `annual_exposure` and assume an equal exposure value of 1 for all observations. Below is a list of all features available in the insurance dataset:

```
'data.frame':   99918 obs. of  31 variables:
 $ pol_bonus       : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.64 0.5 ...
 $ pol_coverage    : Factor w/ 4 levels "Maxi","Median1",..: 1 1 1 3 1 2 1 1 3 1 ...
 $ pol_duration    : int  29 3 2 22 16 5 5 2 5 26 ...
 $ pol_sit_duration: int  9 1 2 1 4 1 3 2 1 6 ...
 $ pol_pay_freq    : Factor w/ 4 levels "Biannual","Monthly",..: 1 1 4 4 1 2 1 1 2 1 ...
 $ pol_payd        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ pol_usage       : Factor w/ 4 levels "AllTrips","Professional",..: 3 3 4 4 3 4 3 3 4 4
                    ...
$ pol_insee_code  : Factor w/ 17794 levels "01001","01004",..: 6424 17569 17550 15522
                    6911 14819 6914 6498 16274 14572 ...
$ drv_drv2        : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 1 1 2 ...
$ drv_age1        : int  85 69 37 81 62 68 77 64 38 59 ...
$ drv_age2        : int  0 0 0 21 68 0 0 0 0 33 ...
$ drv_sex1        : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 2 2 2 2 ...
$ drv_sex2        : Factor w/ 3 levels "","F","M": 1 1 1 2 3 1 1 1 1 2 ...
$ drv_age_lic1    : int  62 39 18 54 37 40 55 37 19 41 ...
$ drv_age_lic2    : int  0 0 0 3 48 0 0 0 0 15 ...
$ vh_age          : int  10 4 11 16 11 14 7 11 9 6 ...
$ vh_cyl          : int  1587 2149 1991 1781 1598 1769 1870 1595 1997 1997 ...
$ vh_din          : int  98 170 150 90 108 60 108 101 109 90 ...
$ vh_fuel         : Factor w/ 3 levels "Diesel","Gasoline",..: 2 1 2 2 2 1 1 2 1 1 ...
$ vh_make         : Factor w/ 101 levels "ACL","ALFA ROMEO",..: 71 60 16 98 76 71 76 7
                    71 71 ...
$ vh_model        : Factor w/ 1023 levels "+4","10","100",..: 75 311 1014 514 599 40 599
                    214 76 725 ...
$ vh_sale_begin   : int  10 4 12 18 13 28 10 16 9 9 ...
$ vh_sale_end     : int  9 2 11 15 11 18 6 13 7 7 ...
$ vh_speed        : int  182 229 210 180 195 155 193 191 183 163 ...
$ vh_type         : Factor w/ 2 levels "Commercial","Tourism": 2 2 2 2 2 2 2 2 2 2 ...
$ vh_value        : num  20700 34250 28661 14407 16770 ...
$ vh_weight       : num  1210 1510 1270 1020 1230 ...
$ claim_nb        : num  0 0 0 0 0 0 0 0 1 0 ...
$ claim_amount    : num  NA NA NA NA NA ...
$ vh_makenew      : Factor w/ 18 levels "Group 1","Group 10",..: 2 7 15 15 1 2 1 7 2 2
                    ...
$ annual_exposure : num  1 1 1 1 1 1 1 1 1 1 ...
```

Please note that the accuracy of $f(x)$ does not directly impact the effectiveness of our fooling algorithm. For demonstration purposes, we construct a claim frequency model, denoted as $f(x)$, using the insurance dataset. Besides `annual_exposure` and `claim_nb` as target or offset variables, we have selected 14 features for the model. These include four policy characteristics (`pol_bonus`, `pol_coverage`, `pol_duration`, `pol_sit_duration`), three primary policyholder characteristics (`drv_age1`, `drv_sex1`, `drv_age_lic1`), and seven vehicle characteristics (`vh_age`, `vh_cyl`, `vh_din`, `vh_fuel`, `vh_speed`, `vh_value`, `vh_makenew`). Table 1 lists the variables used in the insurance dataset, with descriptions extracted from Dutang & Charpentier (2019).

A.2 COMPAS Data

As mentioned in the main text, we directly use the COMPAS data from the fairML R package, which follows the preprocessing steps outlined in Komiyama et al. (2018). The dataset comprises

Variable Name	Description
Response Variables	
claim_nb	The claim number.
annual_exposure	The exposure as a fraction of year, which is assumed to be 1 for all observations. Note that this variable is used as an offset for the model.
Explanatory Variables	
pol_bonus	The policy bonus (French no-claim discount).
pol_coverage	The coverage category, including 4 types : Mini, Median1, Median2 and Maxi, in this order.
pol_duration	The policy duration.
pol_sit_duration	The policy current endorsement (situation) duration.
drv_age1	The driver age of the 1st driver. This is a focal variable for adversarial attacks.
drv_sex1	The driver sex of the 1st driver.
drv_age_lic1	The age of the driving license of the 1st driver.
vh_age	The vehicle age.
vh_cyl	The engine cylinder displacement.
vh_din	A representation of the motor power.
vh_fuel	The vehicle fuel type.
vh_speed	The vehicle maximum speed (km/h), as stated by the manufacturer.
vh_value	The vehicle's value (replacement value). This is a focal variable for adversarial attacks.
vh_makenew	The vehicle carmaker, which are reduced from 101 categories to 18 categories by merging those with similar average responses.

Table 1: Variables Used in $f(\mathbf{x})$ for the Insurance Data

5,855 observations. Below is a list of all features available in the COMPAS dataset:

```
'data.frame':   5855 obs. of  16 variables:
 $ age            : num  69 34 24 44 41 39 21 27 23 37 ...
 $ juv_fel_count  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ decile_score   : num  1 3 4 1 6 1 3 4 6 1 ...
 $ juv_misd_count : num  0 0 0 0 0 0 0 0 0 0 ...
 $ juv_other_count: num  0 0 1 0 0 0 0 0 0 0 ...
 $ v_decile_score : num  1 1 3 1 2 1 5 4 4 1 ...
 $ priors_count   : num  0 0 4 0 14 0 1 0 3 0 ...
 $ sex            : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 1 2 2 2 1 ...
 $ two_year_recid : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 1 2 1 2 1 ...
 $ race           : Factor w/ 6 levels "African-American",..: 6 1 1 6 3 3 3 3 1 3 ...
 $ c_jail_in      : num  0.801 0.784 0.791 0.811 0.818 ...
 $ c_jail_out     : num  0.801 0.785 0.791 0.811 0.818 ...
```

```
$ c_offense_date : num  0.801 0.784 0.791 0.811 0.818 ...
$ screening_date : num  0.801 0.785 0.791 0.811 0.818 ...
$ in_custody     : num  0.829 0.784 0.796 0.811 0.821 ...
$ out_custody    : num  0.83 0.785 0.796 0.811 0.823 ...
```

In addition, descriptions for variables in the COMPAS dataset are provided in Table 2, extracted from the fairML R package.

Variable Name	Description
Response Variable	
two_year_recid	A factor with two levels "Yes" and "No" (if the person has recidivated within two years).
Explanatory Variables	
sex	A factor with levels "Female" and "Male". This is a focal variable for adversarial attacks.
race	A factor encoding the race of the person. This is a focal variable for adversarial attacks.
juv_fel_count	The number of juvenile felonies.
decile_score	The decile of the COMPAS score.
juv_misd_count	The number of juvenile misdemeanors.
juv_other_count	The number of prior juvenile convictions that are not considered either felonies or misdemeanors.
v_decile_score	The predicted decile of the COMPAS score.
priors_count	The number of prior crimes committed.
c_jail_in	The date in which the person entered jail (normalized between 0 and 1).
c_jail_out	The date in which the person was released from jail (normalized between 0 and 1).
c_offense_date	The date the offense was committed.
screening_date	The date in which the person was screened (normalized between 0 and 1).
in_custody	The date in which the person was placed in custody (normalized between 0 and 1).
out_custody	The date in which the person was released from custody (normalized between 0 and 1).

Table 2: Variables Used in $f(\mathbf{x})$ for the COMPAS Data

B Correlations of Features within Datasets Used

Our adversarial framework relies on the assumption that the target feature is not independent of other features in the dataset. In this section, we use Spearman's rank correlation to present and examine the correlation matrix for all features used in the auto insurance claims and COMPAS datasets. For nominal variables, we approximate correlations by transforming the data into ordinal

form, reordering categories based on their respective risk levels. This transformation applies to `pol_coverage`, `vh_fuel`, and `vh_makenew` in the insurance data, and `race` in the COMPAS data.

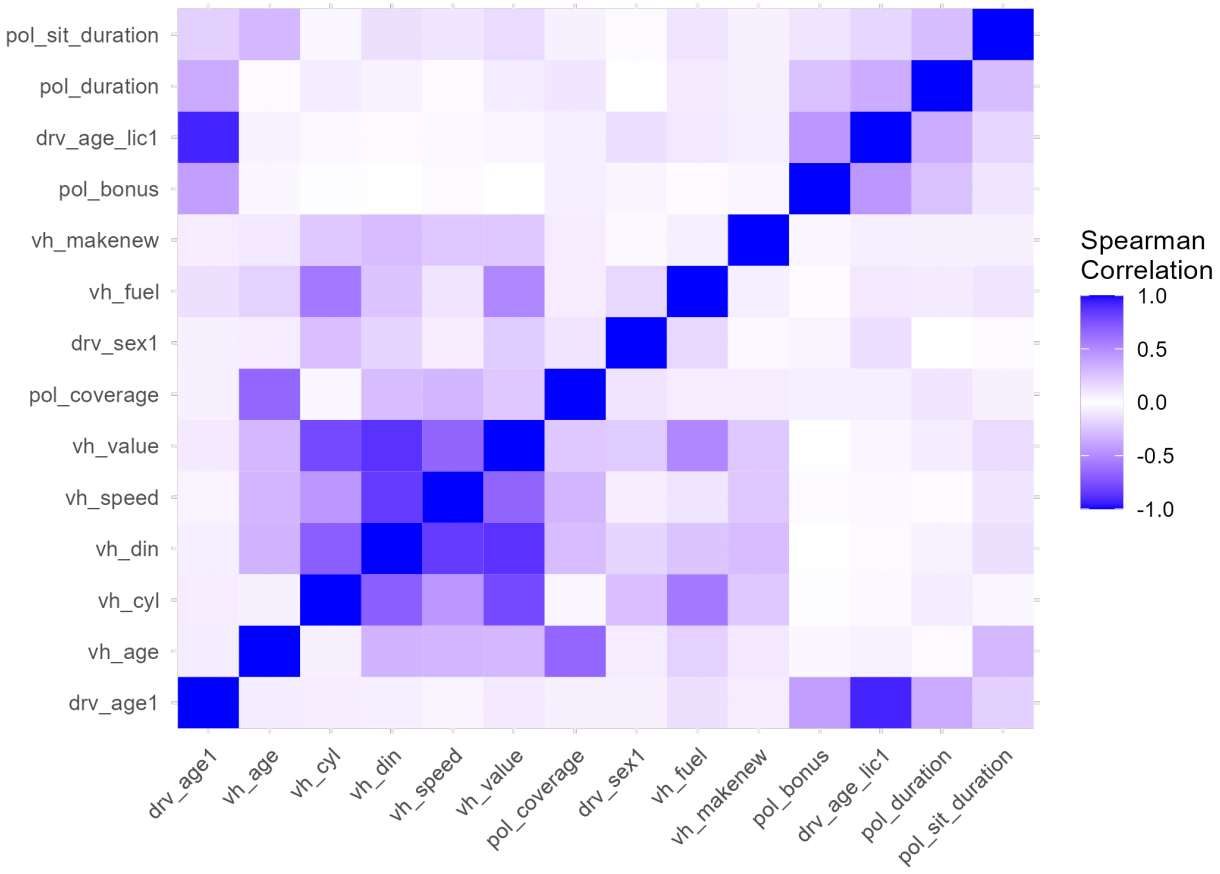

Figure 1: Correlation Matrix for the Insurance Data

As depicted in Figure 1 for the insurance data, the driver's age (`drv_age1`) shows the highest correlation with the age of the driving license (`drv_age_lic1`), followed by the policy bonus (`pol_bonus`) and the policy duration (`pol_duration`). Furthermore, the vehicle characteristic variables exhibit mutual correlations.

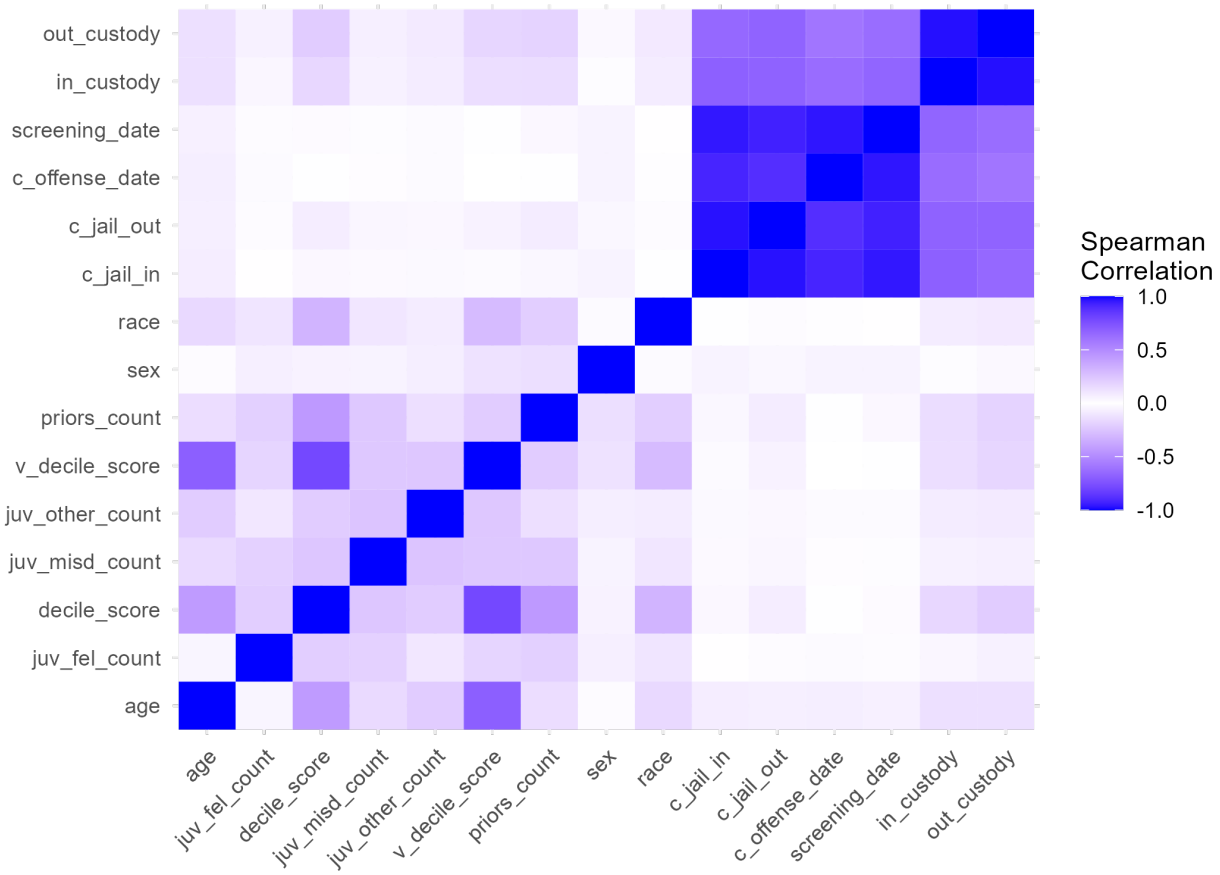

Figure 2: Correlation Matrix for COMPAS Data

For the COMPAS data, as depicted in Figure 2, it is expected that some date-related features exhibit high correlations amongst themselves (`c_jail_in`, `c_jail_out`, `c_offense_date`, `screening_date`, `in_custody`, `out_custody`). The variable `v_decile_score`, a continuous variable containing the predicted decile of the COMPAS score, shows the most significant correlation with age.

C More Implementation Details of Adversarial Framework $a(\mathbf{x})$

In this work, our training data is split into five-folds, and each time, four of the folds are combined as the training data, and the remaining fold is the testing data.

C.1 The Training of $c(\mathbf{x})$

The training process of $c(\mathbf{x})$ involves generating an augmenting sample $\tilde{X}$, which, as outlined in the main text, is drawn randomly and independently among features according to the uniform $\mathbb{P}(\tilde{\mathbf{x}})$. The number of observations in $\tilde{X}$ is 30 times the sample size of the insurance data, and 100 times for both the COMPAS and simulated data. Specifically, for each row in $\tilde{X}$:

- For categorical variables, each feature value $\tilde{x}_j^{(i)}$ is randomly generated from all possible categories of the respective feature.
- For numerical features, each feature value $\tilde{x}_j^{(i)}$ is randomly generated from all unique values observed within the feature in the training data (for instance, the vehicle value in the insurance data).

This approach to generating numerical features is specifically designed to avoid the issue of extrapolation within the range of the training data, ensuring that any unobserved values not present in X are excluded. Consequently, $c(\mathbf{x})$ is constructed to avoid relying on the unique value of a feature for distinguishing between uniform and real data.

C.2 The Selection of Grid Values for PD Plots

For the continuous feature (`vh_value` in the insurance data), the grid values are selected based on quantiles of the feature distribution, instead of using equally spaced points. This approach ensures that the grid points align with the data's distribution. For categorical or discrete features, we employ all available categories or unique values to construct the PD plot.

D Training of $f(\mathbf{x})$, $c(\mathbf{x})$ and $g(\mathbf{x})$ Models

D.1 Training of XGBoost Models

Table 3: XGBoost Hyperparameter Optimization

Hyperparameter	Values	Optimal Value
max_depth	2, 3, 4, 5	3
eta	0.01, 0.05, 0.1	0.01
gamma	0, 0.1, 0.5	0
subsample	0.6, 0.8, 1	0.8
colsample_bytree	0.6, 0.8, 1	0.8
min_child_weight	1, 3, 5	1

For the claim count model $f(\mathbf{x})$ applied to insurance data, we fit the XGBoost model, using the `xgboost` package (Chen et al., 2022) in R. Hyperparameter tuning was performed via grid search, with the optimal parameters detailed in Table 3. It is important to note that the accuracy of $f(\mathbf{x})$ does not directly impact the effectiveness of our adversarial framework $a(\mathbf{x})$.

D.2 Training of Neural Network Models

Table 4: Summary of Neural Network Architectures

Dataset	**Model**	**Structure (Nodes per Layer)**	**Techniques**	**Activation Functions**
Insurance	$c(\mathbf{x})$	[40, 20, 10, 1]	Batch Norm, Dropout (0.2)	ReLU (Output: Sigmoid)
Insurance	$g(\mathbf{x})$	[40, 20, 10, 3]	Batch Norm, Dropout (0.2)	ReLU (Output: Softmax)
COMPAS	$c(\mathbf{x})$	[40, 10, 1]	Batch Norm, Dropout (0.2)	ReLU (Output: Sigmoid)
COMPAS	$g(\mathbf{x})$	[40, 10, 3]	Batch Norm, Dropout (0.2)	ReLU (Output: Softmax)
COMPAS	$f(\mathbf{x})$	[40, 10, 1]	Batch Norm, Dropout (0.2)	ReLU (Output: Sigmoid)
Simulated	$c(\mathbf{x})$	[20, 10, 1]	Batch Norm, Dropout (0.2)	ReLU (Output: Sigmoid)
Simulated	$g(\mathbf{x})$	[20, 10, 3]	Batch Norm, Dropout (0.2)	ReLU (Output: Softmax)
Simulated	$f(\mathbf{x})$	[20, 10, 1]	None	ReLU (Output: Linear)

The neural network models were trained using the `keras` and `tensorflow` packages in R. Training of each neural network model starts with a smaller, commonly used model architecture, gradually incorporating additional features and enhancements. For further tuning strategies, we direct interested readers to Godbole et al. (2023). A summary of the neural network models' architecture is presented in Table 4. Moreover, techniques such as early stopping and learning rate scheduling are employed to mitigate overfitting and enhance model performance.

E Other Experimental Results

E.1 Experimental Results Using Algorithm 1

In this appendix, we present the experimental results for the insurance and COMPAS datasets using Algorithm 1 in Figures 3 to 6. Generally, Algorithm 2 does not show significantly poorer performance in terms of stability for the chosen fold and threshold (i.e., holding out fold 1 and setting the threshold of $c(\mathbf{x})$ to 0.5). The marginally better performance of Algorithm 2 on the COMPAS data is likely due to the errors in $\hat{g}_c(\mathbf{x})$ accidentally compensating for the errors in estimating $\hat{\rho}_j$, $\hat{\lambda}_j$ and $\hat{\gamma}_j$.

E.2 Permutation Feature Importance for Insurance and COMPAS Data

In this appendix, we present the permutation feature importance of our fooling algorithm for the insurance and COMPAS datasets in Figures 7 and 8.

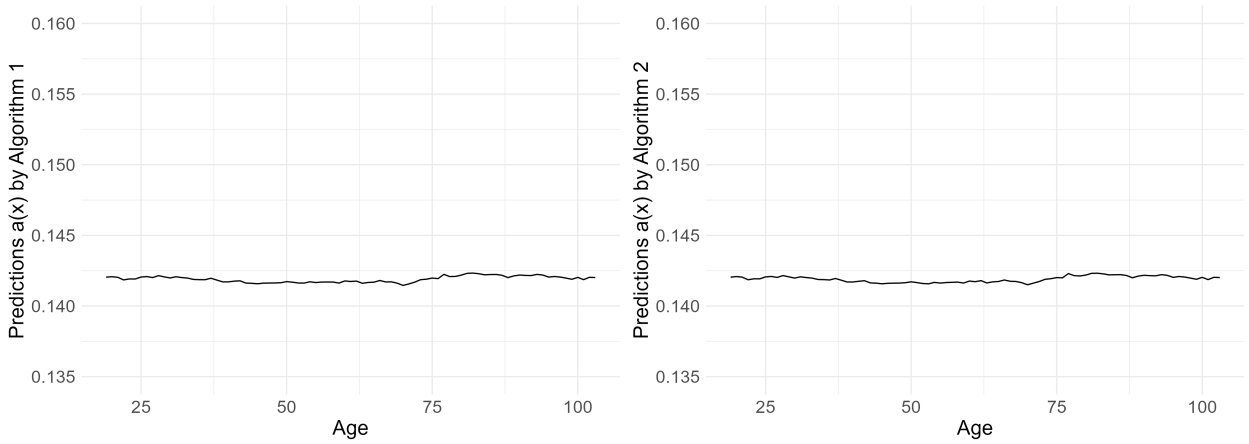

Figure 3: PD Plots for Age after Attack Using Algorithm 1 (Left) and Algorithm 2 (Right), pg17 Data

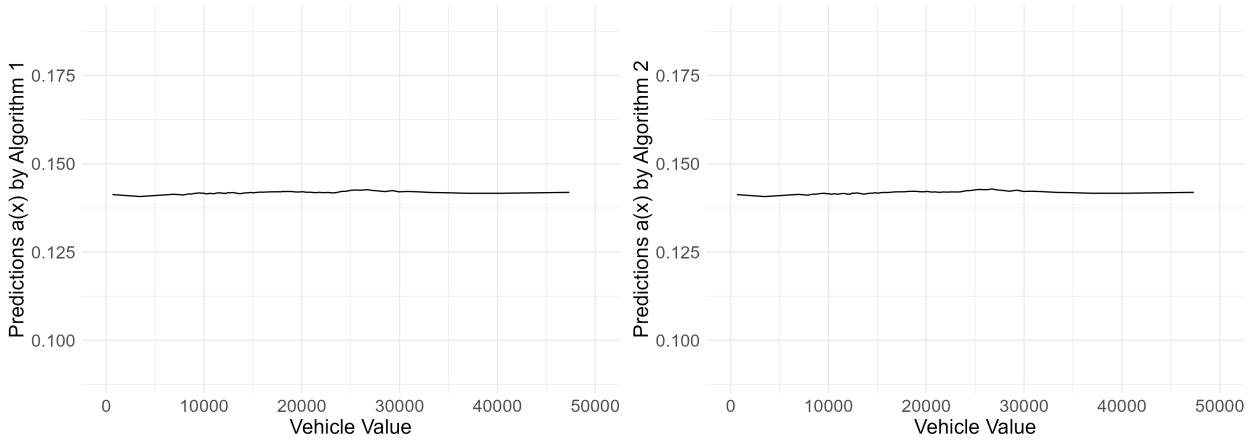

Figure 4: PD Plots for Vehicle Value after Attack Using Algorithm 1 (Left) and Algorithm 2 (Right), pg17 Data

F Alternatives to PD Plots

In this appendix, we present several alternatives to PD plots aimed at addressing their limitations.

Individual Conditional Expectation (ICE) plots: ICE plots (Goldstein et al., 2015) offer a more detailed view by breaking down the averages of PD plots into individual observations. Each ICE curve represents the conditional relationship between a range of X_j values (where $|S| = 1$) and $\hat{f}$ while keeping other features X_C fixed. Mathematically, it is defined as:

$$\text{ICE}_j^{(i)}(x) = \hat{f}(x_j^{(i)} = x, \mathbf{x}_C^{(i)}) \tag{F.1}$$

It should be noted that a PD curve ($\text{PD}_j(x)$) is essentially the average of n ICE curves. ICE plots are useful for identifying heterogeneity in relationships, demonstrating how different subgroups or individual data points respond to changes in the predictor variable. Moreover, similar ICE curves

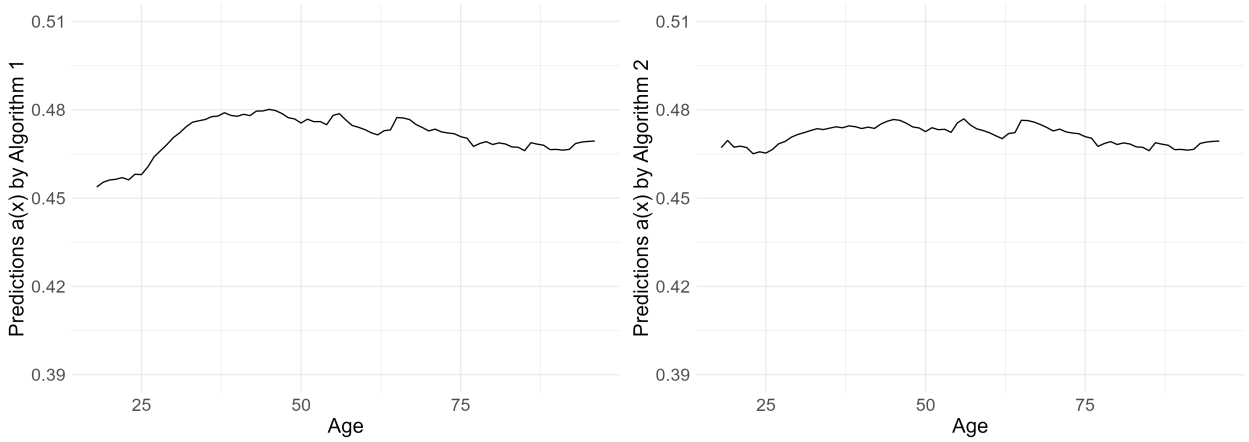

Figure 5: PD Plots for Age after Attack Using Algorithm 1 (Left) and Algorithm 2 (Right), COMPAS Data

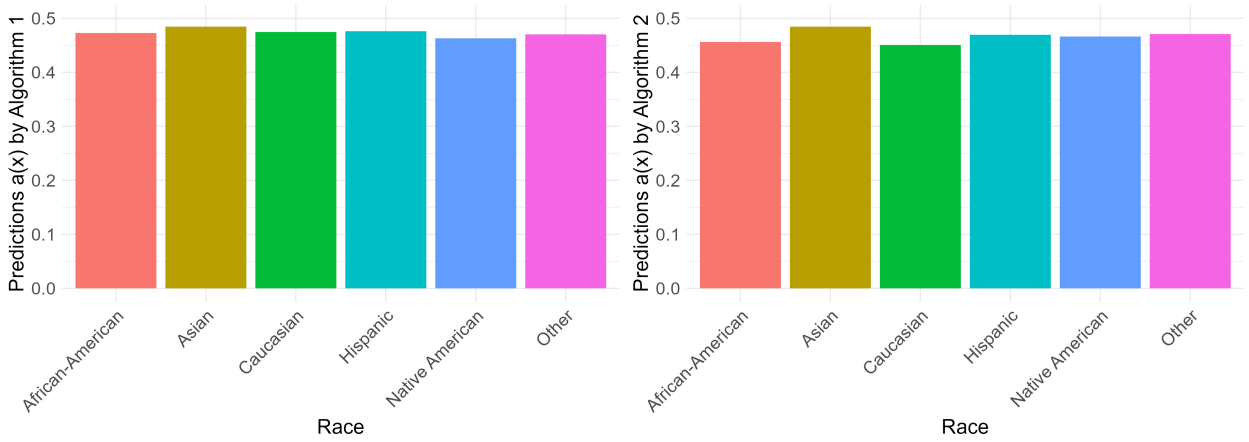

Figure 6: PD Plots for Race after Attack Using Algorithm 1 (Left) and Algorithm 2 (Right), COMPAS Data

can be grouped to avoid ICE plots overplotting and offer a version of regional PD plots, thereby facilitating the identification of regions with less confounded feature effects (Britton, 2019; Zhang et al., 2021; Herbinger et al., 2022).

Accumulated Local Effect (ALE) plots: ALE plots (Apley & Zhu, 2020) average the local changes in the predictions and accumulate them across a grid of intervals. Similar to M-plots, ALE plots use conditional distributions $\mathbb{P}(X_C|X_S = x_S)$ rather than marginal distributions $\mathbb{P}(X_C)$ to mitigate the extrapolation issues present in PD plots. However, ALE plots can avoid the omitted nuisance variable bias, a limitation affecting the usefulness of M-plots. It is important to note that the interpretations of ALE plots are only locally valid within each interval.

In addition, compared to the recovery properties of PD plots, ALE plots adhere to additive $\hat{f}$ regardless of the correlation between X_S and X_C, and they adhere to multiplicative $\hat{f}$ when X_S and X_C are independent (Apley & Zhu, 2020).

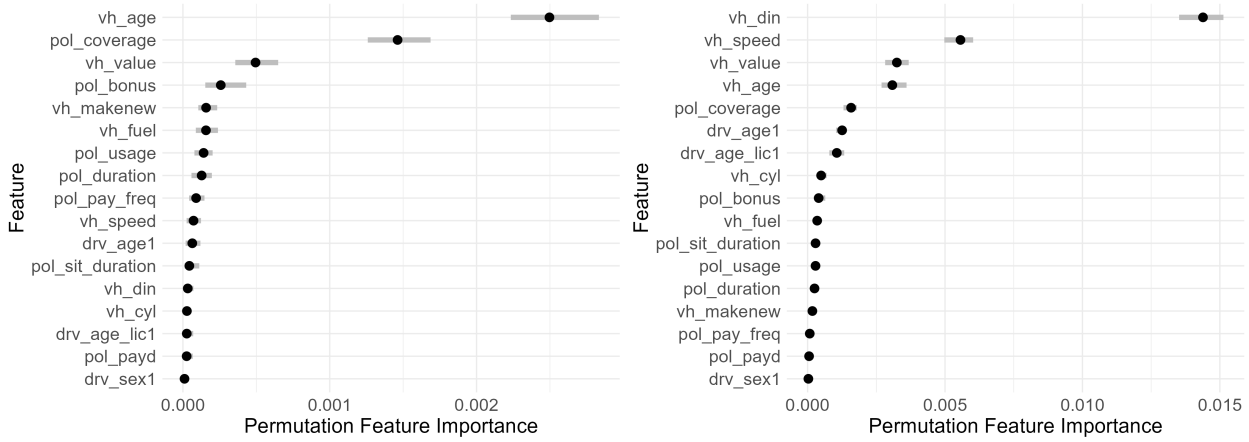

Figure 7: Permutation Feature Importance for $f(\mathbf{x})$ before Attack (Left) and $a(\mathbf{x})$ after Attack (Right), Insurance Data. The grey lines represent the range from the 10th to the 90th percentile of PFI estimates across fifty runs, with median values indicated by black dots.

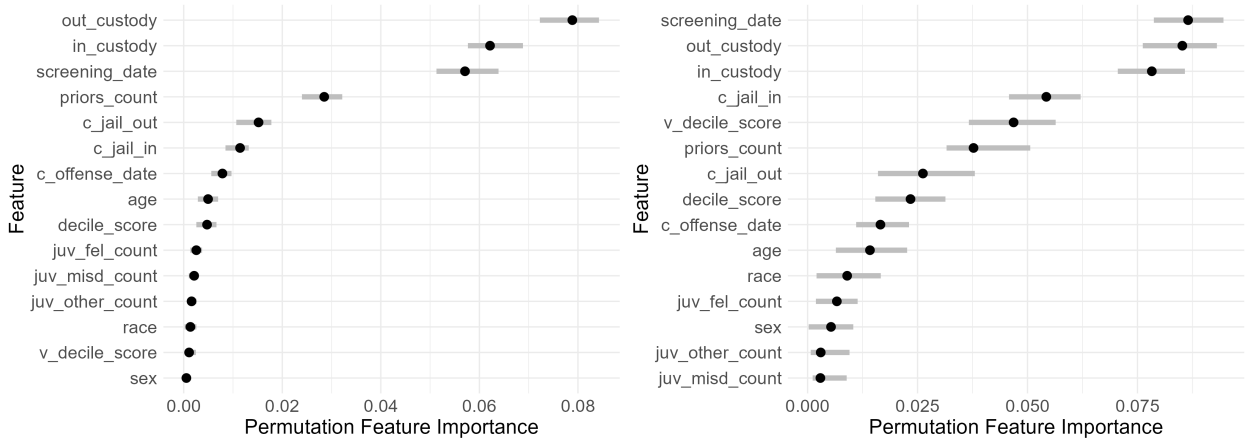

Figure 8: Permutation Feature Importance for $f(\mathbf{x})$ before Attack (Left) and $a(\mathbf{x})$ after Attack (Right), COMPAS Data. The grey lines represent the range from the 10th to the 90th percentile of PFI estimates across fifty runs, with median values indicated by black dots.

Functional ANOVA Decomposition: The concept of **functional ANOVA decomposition** as introduced by (Hooker, 2007) is another avenue associated with PD plots. The PD function's counterparts can be derived by estimating $f_j(x_j)$ and $f_{-j}(\mathbf{x}_{-j})$ to minimize the following expression:

$$\int (f(\mathbf{x}) - f_j(x_j) - f_{-j}(\mathbf{x}_{-j}))^2 p(\mathbf{x}) d\mathbf{x} \tag{F.2}$$

Here, $f_j(x_j)$ represents the individual effect for feature X_j, $f_{-j}(\mathbf{x}_{-j})$ denotes an unknown function for all features except feature X_j, and $p(\mathbf{x})$ represents the feature distribution. See also Hiabu et al. (2023). It is worth noting that accurate estimation can be challenging, especially in high-dimensional settings.

Confidence Intervals of PD Plots: Alternatively, we can derive the **confidence intervals of**

PD plots on bootstrap samples (for example, see Cafri & Bailey (2016)). Molnar et al. (2021) suggested computing the confidence bands for PD plots by refitting models multiple times, thereby accounting for model variance.

References

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *82*(4), 1059–1086.

Britton, M. (2019). Vine: Visualizing statistical interactions in black box models. *arXiv preprint arXiv:1904.00561*.

Cafri, G., & Bailey, B. A. (2016). Understanding variable effects from black box prediction: Quantifying effects in tree ensembles using partial dependence. *Journal of Data Science*, *14*(1), 67–95.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., ... others (2022). *Xgboost: extreme gradient boosting. r package version 1.6. 0.1.*

Dutang, C., & Charpentier, A. (2019). Casdatasets: insurance datasets. *R package version*, 1–0.

Godbole, V., Dahl, G. E., Gilmer, J., Shallue, C. J., & Nado, Z. (2023). *Deep learning tuning playbook.* Retrieved from `http://github.com/google-research/tuning_playbook` (Version 1.0)

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, *24*(1), 44–65.

Havrylenko, Y., & Heger, J. (2022). Detection of interacting variables for generalized linear models via neural networks. *arXiv preprint arXiv:2209.08030*.

Herbinger, J., Bischl, B., & Casalicchio, G. (2022). Repid: Regional effect plots with implicit interaction detection. In *International conference on artificial intelligence and statistics* (pp. 10209–10233).

Hiabu, M., Meyer, J. T., & Wright, M. N. (2023). Unifying local and global model explanations by functional decomposition of low dimensional structures. In *International conference on artificial intelligence and statistics* (pp. 7040–7060).

Hooker, G. (2007). Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, *16*(3), 709–732.

Komiyama, J., Takeda, A., Honda, J., & Shimao, H. (2018). Nonconvex optimization for regression with fairness constraints. In *International conference on machine learning* (pp. 2737–2746).

Molnar, C., Freiesleben, T., König, G., Casalicchio, G., Wright, M. N., & Bischl, B. (2021). Relating the partial dependence plot and permutation feature importance to the data generating process. *arXiv preprint arXiv:2109.01433*.

Zhang, X., Wang, Y., & Li, Z. (2021). Interpreting the black box of supervised learning models: Visualizing the impacts of features on prediction. *Applied Intelligence*, *51*(10), 7151–7165.