Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Bachelor's diploma thesis

in the field of study Data Science

Adversarial attacks on Explainable AI methods

Hubert Baniecki

student record book number 290761

Wojciech Kretowicz

student record book number 291127

thesis supervisor

dr hab. inż. Przemysław Biecek, prof. uczelni

WARSAW 2021

....

supervisor's signature

....

authors' signatures

Abstract

Adversarial attacks on Explainable AI methods

Explainability and transparency of predictive models became a "must-have" feature, especially in financial or medical applications. So-called "right to explanation" is enforced in the growing number of strategies and regulations related to artificial intelligence (AI). However, it turns out that Explainable AI methods are not as secure as we thought. This thesis presents new techniques for attacking post-hoc model-agnostic explanations, namely Partial Dependence Plots (PDP) and Accumulated Local Effects (ALE). We showcase that these visual explanations can present various outcomes determined in an adversarial manner, which causes a lack of trust, even the potential of a security breach. The strategy is to poison the data used to calculate explanations, which leads to a visual change in their results. We implement these algorithms using Python language and showcase their use on a real predictive task.

Further, we perform experiments and discuss explanations of which state-of-the-art machine learning models demonstrate robustness against the data shift, and how model complexity correlates with a probability of successful attack. Our results aim to progress the responsible approach to AI. To the best of our knowledge, we are first to consider attacks on PDP and ALE methods. Additionally, this is the first work performing attacks on model explanations using a genetic algorithm, which can be generalized in a model-agnostic and explanation-agnostic manner.

Keywords: explainable AI, interpretable machine learning, adversarial attack, responsible machine learning, security

Streszczenie

Ataki na metody wyjaśnialnej SI

Wyjaśnialność i transparentność modeli predykcyjnych stała się cechą obowiązkową, zwłaszcza w zastosowaniach finansowych lub medycznych. W coraz większej liczbie strategii i regulacji związanych z sztuczną inteligencją (SI) wprowadza się tak zwane "prawo do wyjaśnienia". Okazuje się jednak, że metody wyjaśnialnej SI nie są tak bezpieczne, jak sądziliśmy. Niniejsza praca przedstawia nowe techniki atakowania wyjaśnień *post-hoc* i niezależnych od struktury modelu, mianowicie *Partial Dependence Plots* (PDP) oraz *Accumulated Local Effects* (ALE). Pokazujemy, że te wizualne wyjaśnienia mogą przedstawiać różne wyniki określone w sposób kontradyktoryjny, co powoduje brak zaufania, a nawet potencjalne naruszenie bezpieczeństwa. Strategia polega na zatruciu danych wykorzystywanych do obliczania wyjaśnień, co prowadzi do wizualnej zmiany ich wyników. Implementujemy te algorytmy przy użyciu języka Python i prezentujemy ich zastosowanie w prawdziwym zadaniu predykcyjnym.

Ponadto przeprowadzamy eksperymenty i omawiamy, wyjaśnienia których znanych modeli uczenia maszynowego wykazują odporność na zmiany w danych oraz jak złożoność modelu koreluje z prawdopodobieństwem udanego ataku. Nasze wyniki mają na celu rozwinięcie odpowiedzialnego podejścia do sztucznej inteligencji. Zgodnie z naszą najlepszą wiedzą jako pierwsi rozważamy ataki na metody PDP i ALE. Dodatkowo, jest to pierwsza praca przeprowadzająca ataki na wyjaśnienia modeli przy użyciu algorytmu genetycznego, które można uogólniać w sposób niezależny od modelu i niezależny od wyjaśniania.

Słowa kluczowe: wyjaśnialna SI, interpretowalne uczenie maszynowe, *adversarial attack*, odpowiedzialne uczenie maszynowe, bezpieczeństwo

Warsaw, ...

Declaration

I hereby declare that I wrote my part of the Engineering thesis (according to the division of work described in the "Division of Work" Chapter at the end of the thesis) entitled "Adversarial attacks on Explainable AI methods" on my own (apart from the recognized reference), under the guidance of the thesis supervisor dr hab. inż. Przemysław Biecek, prof. uczelni.

.....

Hubert Baniecki

Warsaw, ...

Declaration

I hereby declare that I wrote my part of the Engineering thesis (according to the division of work described in the "Division of Work" Chapter at the end of the thesis) entitled "Adversarial attacks on Explainable AI methods" on my own (apart from the recognized reference), under the guidance of the thesis supervisor dr hab. inż. Przemysław Biecek, prof. uczelni.

.....

Wojciech Kretowicz

Contents

1. Introduction

Machine learning models support decision systems, achieve human-like performance in various tasks, or even surpass humans [Bochkovskiy et al., 2020; Brown et al., 2020]. The last step is primarily attributed to growing neural network architectures becoming artificial intelligence (AI). Although these became state-of-the-art solutions to many predictive problems, there is an emerging discussion on the underspecification of such methods which exhibits differing model behaviour in training and practical setting [D'Amour et al., 2020]. This is especially crucial when proper accountability for machine learning predictive systems is required by the domain.

Historically, Lipton [2018] discusses the interpretability of models which needs a proper formulation for future research to be in the right direction. Key desiderata to mention are transferability of machine learning to different, possibly drifting, data; casualty and informativeness which provide discovery in the research or data analysis setting; finally, fair and ethical decision-making, which is an emerging topic for practitioners and regulators. Miller [2019] broadens the view of AI explainability by providing key concepts from the human-computer interaction standpoint and social sciences. There is also a discussion on the semantics of interpretability and explainability terms; for the context of this work, explainability is viewed as post-hoc interpretability. Rudin [2019] argues that that black-box models, with their explanations, will be superseded by interpretable ones. Although diverse machine learning algorithms are recently proposed to solve this debate [Alvarez Melis and Jaakkola, 2018; Angelov and Soares, 2020; Arik and Pfister, 2020; Chen et al., 2020; Konstantinov and Utkin, 2020; Sudjianto et al., 2020], they are yet not proven to be superior. Living with black-boxes, several explainability methods were presented to help us understand models' behaviour [Friedman, 2001; Ribeiro et al., 2016; Lundberg and Lee, 2017; Adadi and Berrada, 2018; Apley and Zhu, 2020], moreover precisely for neural networks [Bach et al., 2015; Shrikumar et al., 2017; Sundararajan et al., 2017; Ancona et al., 2018; Kim et al., 2018; Alber et al., 2019]. These are widely used in practice through their (often estimation-based) implementations available to machine learning practitioners in various software contributions [Biecek, 2018; Alber et al., 2019; Arya et al., 2020].

1.1. Motivation

The summary of the Explainable AI (XAI) domain we adhere to presented by Barredo Arrieta et al. [2020] formulates several challenges for future research; namely developing metrics and evaluating explainability methods, also reassuring security concerns and the corresponding adversary in the AI setting. All these become crucial when using model explanations in the Data Science practice to ensure proper understanding of black-box machine learning; thus, facilitate rationale explanation and responsible decision-making [Gill et al., 2020; Hancox-Li, 2020]. Several studies evaluate model explanations [Adebayo et al., 2018; Hooker et al., 2019; Adebayo et al., 2020; Bhatt et al., 2020; Warnecke et al., 2020] showcasing their various flaws from which we perceive the existing robustness gap; one can call it a *security breach*. Apart from promoting wrong explanations, this phenomenon can be exploited to produce adversarial attacks on model explanations and achieve manipulation, *fooling*. Models require proper performance tests, so will model explanations in the near future. We aim to perform attacks on XAI methods to check their robustness and showcase the potential of adversarial usage. Figure 1.1 shows the rising popularity of Partial Dependence Plots [Friedman, 2001] targeted by this work, which is an additional quantitive motivational factor.

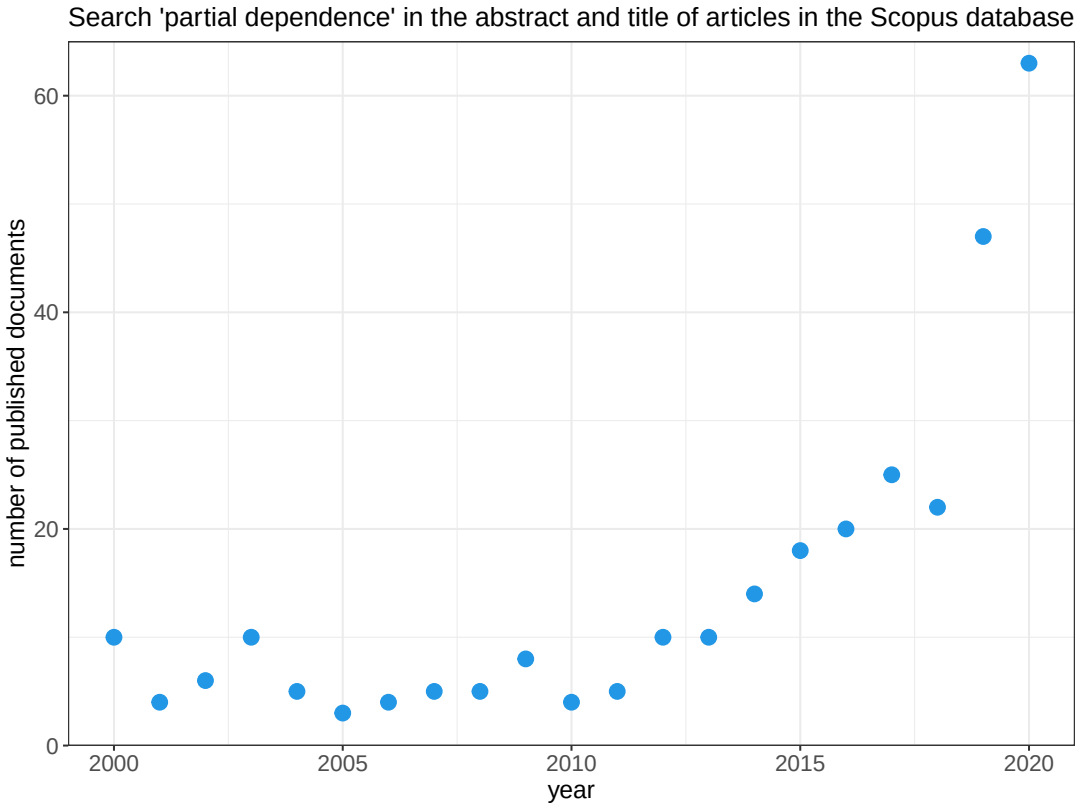

Figure 1.1: Results of the "`TITLE-ABS-KEY ("partial dependence") AND PUBYEAR < 2021`" search in the Scopus database `https://www.scopus.com/`.

1.2. Contribution

In this thesis, we introduce two algorithmic ways of fooling model-agnostic, post-hoc explanation methods for *global-level* understanding. These operate on representative subsets of data, sometimes whole training dataset, or validation dataset with similar variable distributions; thus, we investigate how data shift affects the visual result of explanation. We target Partial Dependence Plots (PDP) [Friedman, 2001] and Accumulated Local Effects (ALE) [Apley and Zhu, 2020] to showcase the potential of a security breach, moreover provide sanity checks for their future use. Further, we contribute a novel concept of using a genetic algorithm for such adversary. This allows for a convenient generalization of the attacks in a model-agnostic and explanation-agnostic manner, which is not the case for most of the related works. The main implication of the proposed methods, changing the result of explanation through data poisoning, is showcased on an accessible machine learning predictive task from the medical domain. Finally, we provide an evaluation of the constructed attacks on PDP and ALE in experiments that aim to answer questions like which models provide robustness to such adversary, and how model complexity correlates with a probability of successful attack.

Next section provides a brief summary of existing adversarial attacks on Explainable AI methods. The work is organised into four main Chapters. Chapter 2 introduces several theoretical concepts which are further assumed as acknowledged. In Chapter 3 we present attacks on PDP and ALE methods. Chapter 4 describes a practical implementation of the introduced methods in the Python package with an example of usage. Finally, Chapter 5 provides quantitive experiments using the attacks.

1.3. Related work

In the literature, there is a considerate amount of attacks for model explanations specific to the deep neural network domain [Dombrowski et al., 2019; Ghorbani et al., 2019; Heo et al., 2019; Kindermans et al., 2019; Zhang et al., 2020]. At their core, they provide various algorithms for fooling neural network explainability, mainly of image-based predictions. These are commonly named as saliency maps [Simonyan et al., 2014], where each model input is given its attribution to the prediction [Bach et al., 2015; Sundararajan et al., 2017; Shrikumar et al., 2017; Selvaraju et al., 2020], and are quite distinct from the realm of PDP and ALE methods.

When considering an explanation as a function of model and data, there is a possibility to change one of these variables to achieve a different result. Heo et al. [2019] propose fine-

tuning of a neural network to undermine its explainabiltiy capabilities. The assumption is to alter model's parameters without a drop in performance, which is achieved with an objective function minimizing the distance between the result of explanations and an arbitrally set target. Another idea is to manipulate explanations via data change. Dombrowski et al. [2019] proposed an algorithm for visual explanation manipulation using gradient-based data perturbations which our work originates from.

In contrast, we investigate the realm of machine learning predictive models trained on tabular data (including neural networks). Slack et al. [2020] contributed adversarial attacks on post-hoc, model-agnostic explanation methods for *local-level* understanding; namely LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017]. The proposed framework provides a way to construct a biased classifier with safe explanations of models' individual predictions. Since we focus on global-level explanations; instead, the results will modify a view of overall models' behaviour, not specific to a single data point or image. Lakkaraju and Bastani [2020] conducted a thought-provoking study on misleading effects of manipulated model explanations which provides arguments for why such research becomes crucial to achieve responsibility in AI use. Rieger and Hansen [2019] present a defense strategy against the attack via data change of Dombrowski et al. [2019] The main idea is to aggregate various model explanations, which produces robust results. The simple strategy doesn't change the model and is computationally inexpensive.

Robustness of neural networks became a crucial factor in nowadays reserach, as we want to trust black-box models and extend their use to more sensitive tasks [Boopathy et al., 2020; Wang et al., 2020]. Further related are studies on security breach in remote explainability [Merrer and Trédan, 2020], and fooling of fairness methods [Fukuchi et al., 2020].

2. Theory

2.1. Explainable AI methods: PDP and ALE

In this thesis, we target two popular Explainable AI methods which at its core present expected value of models' predictions as a function of a selected variable from the model. First, PDP introduced by Friedman [2001], show the expected value fixed over the marginal joint distribution of other variables. These values can be easily estimated and are widely incorporated into various tools for model explainability [Greenwell, 2017; Molnar et al., 2018; SauceCat, 2018; Baniecki and Biecek, 2019; Nori et al., 2019; Baniecki et al., 2020]. Second, ALE introduced by Apley and Zhu [2020], show the expected value fixed over the conditional joint distribution of other variables, additionally accumulated on the fixed range. Such a construct reassures to grasp only the local effects of a given variable and should avoid taking into account interactions between variables in the model.

These theoretical explanations have their practical estimators used to compute the results, later visualized as a line graph showing the expected prediction for a given variable. With that in mind, we refer to their definitions from the *Explanatory Model Analysis* book [Biecek and Burzykowski, 2021]. Preliminary definitions 2.1 and 2.2 are followed by the definitions of model explanations 2.3 and 2.4 with the their corresponding estimators (Equations 2.1 and 2.2).

Definition 2.1. By $X^{c|=z}$ we denote random vector X or dataset X, where $c-th$ variable is replaced by value z.

Definition 2.2. By X_{-c} we denote distribution of random vector X where $c-th$ variable is set to constant.

Definition 2.3 (PDP). Partial Dependence Plot for model f and variable c in a random vector X is defined as follows:

$$PDP_c(z) = E_{X_{-c}}\left[f(X^{c|=z})\right].$$

Definition 2.3 says that Partial Dependence Plot in point z is the average prediction of the model f when the $c-th$ variable is set to constant value z. This is the theoretical concept, while in practice, we use an estimator presented by Definition 2.1.

A standard estimator of Partial Dependence Plot is folowing formula:

$$\widehat{PDP}_c(z) = \frac{1}{n}\sum_{i=1}^{n} f\left(x_i^{c|=z}\right). \tag{2.1}$$

Definition 2.4 (ALE). Accumulated Local Effects for model f and variable c in a random vector X is defined as follows:

$$ALE_c(z) = \int_{z_0}^{z}\left(E_{X_{-c}}\left[\left\{\frac{\partial f(u)}{\partial u}\right\}_{u=X^{c|=z}}\right]\right)dv + c.$$

Definition 2.4 says that Accumulated Local Effects is an integral of expected derivatives of a function with respect to the explained variable, in a point where the $c-th$ variable is set to constant value z. Thus, it is an accumulation of average derivates over explained variable. In practice, we use an estimator presented by Definition 2.2.

A standard estimator of Accumulated Local Effects is following formula:

$$\widehat{ALE}_c(z) = \sum_{k=1}^{K}\left[\frac{1}{\sum_l w_l(z_k)}\sum_{i=1}^{N} w_i(z_k)\left\{f\left(X_i^{c|=z_k}\right) - f\left(X_i^{c|=z_k-\Delta}\right)\right\}\right] - \hat{c}. \tag{2.2}$$

2.2. Genetic algorithm

Genetic algorithms are widely used as an optimization tool [Wright, 1991; Elbeltagi et al., 2005]. We define an individual representing one solution to the problem. Multiple individuals make up the population, where they crossover and mutate in each iteration of the algorithm. After these operators, each individual is evaluated, and selection is performed depending on the scores. The approach presented in Algorithm 1 can be applied to various domains, as heuristics allows to formulate the problem, individuals and scoring function freely. We use it to optimize a loss function in a genetic-based attack introduced in Section 3.2.

ALGORITHM 1: Genetic algorithm

```
initialize the population
while not termination condition do
    crossover
    mutation
    evaluation
    selection
end
select the best individual
```

2.3. Gradient descent

Gradient descent is a generic iterative algorithm used for optimization, where a function that is optimized has to be differentiable. In its simplest version, this algorithm makes a step by moving in a direction opposite to the gradient scaled by the learning rate. If L denotes the function to minimize, α the learning rate, w stands for arguments of the function L, and t denotes the previous iteration, the next step is defined by the equation:

$$w^{(t+1)} := w^{(t)} - \alpha \nabla_w L^{(t)}.$$

We use gradient descent to optimize a loss function in a gradient-based attack in Section 3.3.

2.4. Adam optimizer

Adaptive moment estimation (Adam) [Kingma and Ba, 2015] is an extension to the gradient descent with an adaptative learning rate. It achieves faster convergence compared to the simplest gradient descent.

Adam is used in stochastic optimization thus it considers a gradient calculated at some step as a random variable. It replaces the usage of the gradient with momentum scaled by the second moments of the gradient. Thus if the algorithm makes few steps in a similar direction, it gains a momentum that speeds up model training. Adam optimizer has three more parameters: β_1 controls the momentum, β_2 controls second moments, and ϵ is used to omit the potential numerical errors. We use symbols from the previous section; the following equations define one step of the algorithm:

$$m^{(t+1)} := \beta_1 m^{(t)} + (1 - \beta_1) \nabla_w L^{(t)}, \quad v^{(t+1)} := \beta_2 v^{(t)} + (1 - \beta_2) \left(\nabla_w L^{(t)} \right)^2,$$

$$\hat{m}^{(t+1)} := \frac{m^{(t+1)}}{1 - \beta_1}, \quad \hat{v}^{(t+1)} := \frac{v^{(t+1)}}{1 - \beta_2},$$

$$w^{(t+1)} := w^{(t)} - \alpha \frac{\hat{m}^{(t+1)}}{\sqrt{\hat{v}^{(t+1)}} + \epsilon},$$

where $m^{(t)}$ is a biased estimate of the first moment of the gradient at step t, $\hat{m}^{(t)}$ is a biased corrected $m^{(t)}$, $v^{(t)}$ is a biased estimate of the second moment of the gradient, and $\hat{v}^{(t)}$ is a biased corrected $v^{(t)}$. We use Adam optimizer to speed up the gradient-based attack later in this work.

3. Attacks on PDP and ALE

We consider global-level model explanations which use a dataset to compute their results; thus, they become a function with a multidimensional domain. For example, if the dataset has n instances and p variables, then PDP and ALE domain consists of $n \cdot p$ dimensions. Moreover, because of the complexity of black-box models, these functions become very complicated, and variable interactions cause unpredictable behaviour.

Figure 3.1 presents the main idea of an adversarial attack on model explanation using data poisoning. We aim to change the underlying dataset used to produce models' explanation in a way so that we achieve a fooling of the visual result. In practice, machine learning practitioners compute such explanations using their estimators where a significant simplification may occur. Thus, small shift of the dataset used to calculate PDP or ALE may lead to unexpected or unintended results.

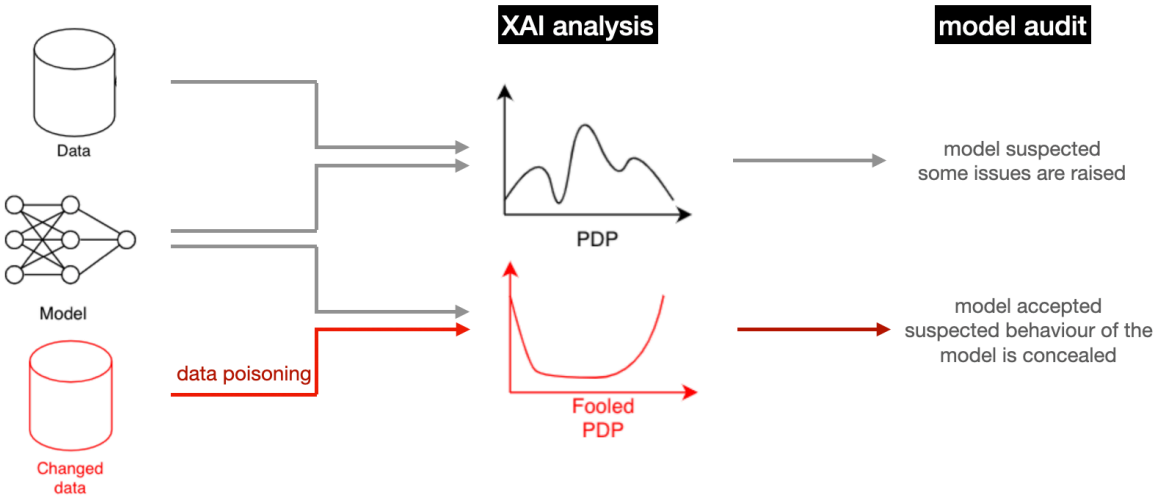

Figure 3.1: Framework for model-agnostic, post-hoc adversarial attacks on global-level Explainable AI (XAI) methods using data poisoning. Red color indicates the possible route, a security breach, which may be used by an attacker to fool the visual result of an explanation. Researchers could use this method to provide a wrong rationale explanation, while auditors could provide false evidence of responsible AI use.

We define attacking the PDP and ALE methods as an optimization problem of a given loss function. This idea originates from the work of Dombrowski et al. [2019], where similar loss

function for manipulation of local-level model explanations for an image-based predictive task was introduced. In this thesis, we introduce a modified loss function that consists of several terms. The first term describes the main goal: the change in the output explanation, while further terms serve as regularization values, e.g. second ensures that the dataset does not derive too far from the original one, and third keeps the mean model prediction stable.

Note that more regularization terms could be added in a customizable manner; it depends on the attacker's priorities. We introduce two algorithms that optimize the introduced loss function:

- **genetic-based**[1] model-agnostic algorithm which does not make any assumption about the model's structure,
- **gradient-based** model-specific algorithm which is designed for models with differentiable outputs, in instance neural networks.

We foresee two possible strategies to perform the fooling:

- **targeted attack** – changes the dataset to achieve the closest explanation result to the predefined desired model explanation,
- **robustness check** – aims to achieve the most distant model explanation from the original one by changing the dataset.

In Sections 3.1-3.3 we formalize the methods needed to perform the introduced attack. Table 3.1 summarises the math notation used in further equations and theorems.

Table 3.1: Summary of math notation.

Sign	Description	Sign	Description
f	model	X'	original (constant) dataset
g, PDP, ALE	explanation	X	changed dataset
c	explained variable	n	number of dataset rows
T	target explanation	p	number of dataset columns
L	loss function	Z	fixed set of c-th variable values
α, β	regularization terms	z	element of Z
$s \in \{t,\ r\}$	strategy of the attack	C	constant variables

[1]For convenience, we shorten the *attack based on the genetic algorithm* phrase to *genetic-based attack*.

3.1. Loss function

The main idea behind attacks is to optimize a loss function. Changed dataset, denoted as $X \in \mathbb{R}^{nxp}$, is an argument of that function (optimal X is a result of an attack). Let $Z \in \mathbb{R}$ be the set of points that are used to calculate PDP or ALE respectively. Let $T : Z \to \mathbb{R}$ be the target explanation and let $g_c^Z : \mathbb{R}^{nxp} \to \mathbb{R}^{|Z|}$ be the actual explanation (PDP or ALE) calculated for points in Z; we write g_c for simiplicity. Finally, let $X' \in \mathbb{R}^{nxp}$ be the original (constant) dataset. We define the following loss function

$$L(X) = L_1^{g,\,s}(X) + \alpha \cdot L_2(X) + \beta \cdot L_3(X), \tag{3.1}$$

where $L_1^{g,\,s}$ is the objective depending on the strategy s, while L_2 and L_3 serve as regularization parameters

$$L_2(X) = \frac{\|X - X'\|^2}{np}, \tag{3.2}$$

$$L_3(X) = \frac{\|f(X) - f(X')\|^2}{n}. \tag{3.3}$$

Depending on the strategy s of the attack we have the following objectives:

- In the **targeted attack** we aim to minimize the distance between the target T and the result of model explanation calculated on the changed dataset

$$L_1^{g,\,t}(X) = \frac{\|g_c(X) - T\|^2}{|Z|}. \tag{3.4}$$

- In the **robustness check** we aim to maximize the distance between the results of model explanation calculated on the original dataset $g_c(X')$ and the changed one

$$L_1^{g,\,r}(X) = -\frac{\|g_c(X) - g_c(X')\|^2}{|Z|}; \tag{3.5}$$

  thus, minus sign is required.

Further, regularization terms α and β are added to the loss function 3.1 to control the algorithm. It is possible to add more loss terms and parameters for various use-cases, which we discuss as future works in Chapter 6.

Since we focus on a specific model-agnostic explanation, we substitute PDP in place of g:

$$L_1^{PDP,\,t}(X) = \frac{\|PDP_c(X) - T\|^2}{|Z|}, \tag{3.6}$$

$$L_1^{PDP,\,r}(X) = \frac{\|PDP_c(X) - PDP_c(X')\|^2}{|Z|}. \tag{3.7}$$

This substitution can be generalized for various global-level model explanations, which rely on using the dataset for computation. For example, to achieve the attack on ALE we substitute

$$L_1^{ALE,\,t}(X) = \frac{\|ALE_c(X) - T\|^2}{|Z|}. \tag{3.8}$$

Note: L_1^g may vary depending on the explanation used, e.g. for PDP it is useful to centre the explanation before calculating the loss, which is the default behaviour in our implementation:

$$L_1^{PDP_{centred},\,r}(X) = \frac{\| \left(PDP_c(X) - \frac{\sum_{z\in Z} PDP_c(X,z)}{|Z|}\right) - \left(PDP_c(X') - \frac{\sum_{z\in Z} PDP_c(X',z)}{|Z|}\right) \|^2}{|Z|}. \tag{3.9}$$

We aim to minimize the loss function (Equation 3.1) with respect to the dataset X used to calculate the explanation. It is possible to set some of the columns in the dataset, or even particular values, as original (constant) values from X'. We define an optional set of such explanatory variables as C; they remain unchanged by the introduced algorithms but contribute to the model predictions and explanations. Note that we never change the explained variable; thus, always $c \in C$.

Next, we introduce two attacks which optimize the loss function to achieve adversary, starting with the one based on the genetic algorithm.

3.2. Genetic-based attack

We propose a novel attack based on the genetic algorithm because it is a simple, yet powerful, method for real parameter optimization [Wright, 1991]. We do not encode genes conventionally, but deliberately use this term to distinguish from other types of evolutionary algorithms [Elbeltagi et al., 2005]. The attack will be invariant to the definition of model and the considered explanations; thus, it becomes model-agnostic and explanation-agnostic. These traits are crucial when working with black-box machine learning because versatile solutions are convenient.

ALGORITHM 2: Genetic-based attack.

Input: f, X', g, c, T, C, α, β
Output: $g_c(X)$, X

initialize P
while *iteration* $<$ *max_iterations* **do**
    crossover phase (Algorithm 3)
    mutation phase (Algorithm 4)
    evaluation phase (Algorithm 5)
    selection phase (Algorithm 6) `// omit selection in the last iteration`
end
$X \leftarrow \text{argmin}_{x \in P} L$

Attacks on PDP and ALE in both strategies include a similar Algorithm 2. The main idea is defining an individual as an instance of the dataset, iteratively perturb its values to achieve the desired explanation target, or perform the robustness check to observe the change. These individuals are initialized with a value of original dataset X' to form a population P. Subsequently, the initialization ends with mutating P using a higher-than-default variance of perturbations. Then, in each iteration, they are randomly crossed, mutated, evaluated with the loss function, and selected based on the evaluation. The algorithm stops after a defined number of repetitions, and the best individual, with its corresponding explanation, is the result.

At no point in the algorithm, the order of dataset rows is changed, as we intend to calculate the distances between two datasets in the loss function; which is crucial because various mutation and crossover implementations are possible. For example, we considered the crossover through exchange of rows between individuals, but it might drastically shift the datasets and move them apart. Additionally, exemplary mutation idea which is out of scope of this work was to add or substract whole numbers from the integer-encoded variables.

We discuss details of the Algorithm 2 in the following sections. Note that the implementation of the attack operates on `3D` matrixes, so that no redundant loops are required in practice. The initialized population moves to the crossover phase.

3.2.1. Crossover

The crossover presented in Algorithm 3 swaps columns between parent individuals to produce new ones. The proportion of the population which becomes parents is parametrized by *crossover_ratio*, and the parent pairs are randomly sampled without replacemen from the subset $P_{crossover\_ratio}$. For each pair, the set of variable columns (full dataset) to swap is randomly selected, and becomes a newly created individual q. Such constructed childs Q are added to the population. The enlarged population moves to the mutation phase.

ALGORITHM 3: Crossover phase.

Input: P, *crossover_ratio*
Output: P

$R \leftarrow P_{crossover\_ratio}$
$Q \leftarrow \{\}$
while *there are individuals in* R **do**
    $n,\ m \leftarrow$ sample a pair of individuals without replacement from R
    $q \leftarrow$ create a new individual from the randomly selected columns of n and m
    $Q \leftarrow Q \cup q$
end
$P \leftarrow P \cup Q$

3.2.2. Mutation

The mutation presented in Algorithm 4 adds a Gaussian noise to the individuals (datasets) using the variables' standard deviations $std(X')$, which results in a changed population P. These standard deviations are scaled by the *std_ratio* parameter to lower the variance of noise. There is a possibility to constraint the changes in the datasets only to the original range of variable values. Then, the potential incorrect values that might occur, are substituted with new ones from the uniform distribution of the range between the original dataset value and the boundaries. It is also practicable to treat chosen elements of the dataset as constant. The main implementation detail of the mutation is that we explicitly omit to change values coded as `0` in the original dataset, as these values might be specially encoded in the data. The mutated population moves to the evaluation phase.

ALGORITHM 4: Mutation phase.

Input: P, X', C, *std_ratio*, *mutation_with_constraints*
Output: P

for *each individual* $m \in P$ **do**
    $\theta \leftarrow noise(std(X') \cdot std\_ratio)$ `// Gaussian noise with mean 0`
    $mask \leftarrow create\_mask(X',\ C)$
    $m \leftarrow m + \theta \cdot mask$
    if *mutation_with_constraints* **then**
        for *each column* $v \in m$ **do**
            find values which are out of the original range $[min(v), max(v)]$
            sample new values from the uniform distribution $U[min(v),\ v\_i]$ or $U[v\_i,\ max(v)]$
            substitute the out-of-range values
        end
    end
end

3.2.3. Evaluation

The evaluation presented in Algorithm 5 uses the loss function described in Section 3.1; thus, depends on the strategy s. For the robustness check r we use the original dataset X' to calculate all loss terms, while for the targeted attack t we require T in $L_1^{g,\,t}$ (Equation 3.4). Genetic algorithms usually maximize the fitness function, but we decided to minimize the loss function so that both introduced attacks are similar. The most time-consuming operation in the whole genetic-based algorithm is performing models' prediction on the whole population of individuals. Thus, we heuristically substitute mean model's prediction $f(X)$ in the L_3 term of the loss function (Equation 3.3) with a mean of the explanation result $g_c(X)$ (expected prediction). Algorithm 5 returns loss values l for each individual which are passed to the selection phase.

ALGORITHM 5: Evaluation phase.

Input: P, X', g, c, s, T, α, β
Output: l `// loss calculated for each individual from the population`

$l \leftarrow \{\}$
for *each individual* $m \in P$ **do**
  `//` $L_1^{g,\,r}$ `uses` X'`, while` $L_1^{g,\,t}$ `uses` T
  $l_m \leftarrow L_1^{g,\,s}(m) \; + \; \alpha \cdot L_2(m) \; + \beta \cdot L_3(m)$
end

3.2.4. Selection

The selection presented in Algorithm 6 uses the rank selection algorithm to reduce the number of individuals to the *pop_count* starting number and ensure attack convergence. Rank selection uses the probability of survival of each individual, which depends on their ranking based on the corresponding loss values l. We added fundamental elitism to the selection algorithm, meaning that in each iteration, we guarantee several best individuals to remain into the next population. This addition ensures that the genetic-based attack's solution quality will not decrease from one iteration to the next. The cycle continues until *max_iter* iterations are reached, and the best individual is selected.

ALGORITHM 6: Selection phase involving rank selection and elitism.

Input: P, l, *pop_count*, *elitism_count*
Output: P

$P' \leftarrow P$ ordered by the values of l
$E \leftarrow$ *elitism_count* best individuals from P
$prob \leftarrow rank(P')$
$P \leftarrow sample(P',\, prob,\, pop\_count)$
$P \leftarrow P \cup E$

We implement this attack in Chapter 4 and evaluate it further in Chapter 5. While we found the genetic-based method a sufficient generalization of our framework, there is a possibility to perform a more efficient attack assuming the knowledge about the model and the explanation.

3.3. Gradient-based attack

Gradient-based methods are state-of-the-art optimization approaches, especially in the domain of neural networks [LeCun et al., 2015]. This attack's main idea is to utilize the gradient descent to optimize the loss function, particularly considering the differentiability of the model's output with respect to the input data. Such assumption allows faster and more accurate convergence of the loss function to the local minima using one of the stochastic optimizers; in this case, Adam [Kingma and Ba, 2015].

Note, that our assumption about the differentiability is with respect to the input data, not in respect to the model's parameters. Although we specifically consider the usage of neural networks because of their strong relation to differentiation, the algorithm's mathematical derivation does not require this type of model.

We shall derive the gradients for attacks on model explanations based on their estimators, not the theoretical definitions. The reason for this is fact that in the theoretical definition of PDP and ALE methods, input data is assumed to be a random variable, making it impossible to calculate the derivative over the input dataset. We also do not want to somehow derive our method directly from the definition, because the estimator is actaully producing results in practice.

Additionally, for PDP, we are considering two different approaches to calculate the first term of the loss function, one requires centring of the model explanation, second does not. Former one forces changes in the shape of the visual model explanation. A loss term without centring may result in the visual explanation shifted upward or downward with shape changed insignificantly.

Further, we provide theorems presenting derivatives of particular terms in the loss function. Sum of these terms weighted by α and β gives gradient of the loss function with respect to all variables in the dataset. Such gradient can be further used in the optimizers.

Assumption: When calculating the derivative regarding PDP_c or ALE_c, we assume that the variable column explained with these methods is constant; thus, we denote it as $\nabla_{X_{-c}}$, where c is the explained variable.

Note: To simplify the notation sometimes we drop index c from PDP_c and ALE_c where context is clear.

Theorem 3.1 (Derivative of $L_1^{PDP,\,t}$ and $L_1^{PDP,\,r}$ (3.6)). Let $f : \mathbb{R}^{nxp} \longrightarrow \mathbb{R}$ represents the differentible function that is explained by PDP. Let Z be the grid of points that are used to calculate PDP. Let $T : Z \to \mathbb{R}$. Finally, let $X' \in \mathbb{R}^{nxp}$ be the original dataset. Then

$$\nabla_{X_{-c}} L_1^{PDP,\,t}(X) = \frac{2}{n|Z|} \sum_{z \in Z} \nabla_{X_{-c}} f(X^{c|=z}) \cdot (PDP_c(X, z) - T(z))$$

$$\nabla_{X_{-c}} L_1^{PDP,\,r}(X) = -\frac{2}{n|Z|} \sum_{z \in Z} \nabla_{X_{-c}} f(X^{c|=z}) \cdot \left(PDP_c(X, z) - PDP_c(X', z)\right).$$

Proof. We derive the formula for the targeted attack, while a formula for the robustness check is derived by analogy. We calculate the derivative with respect to a particular value $X_{i,j}$ in the dataset. Please note, that we want to leave column c intact, so $j \neq c$. We have

$$\frac{\partial L_1^{PDP,\,t}(X)}{\partial X_{i,j}} = \frac{\partial}{\partial X_{i,j}} \frac{1}{|Z|} \sum_{z \in Z} (PDP_c(X, z) - T(z))^2$$

$$= \frac{2}{|Z|} \sum_{z \in Z} (PDP_c(X, z) - T(z)) \frac{1}{n} \sum_{k=1}^{n} \frac{\partial}{\partial X_{i,j}} f(X_k^{c|=z})$$

$$= \frac{2}{n|Z|} \sum_{z \in Z} (PDP_c(X, z) - T(z)) \frac{\partial}{\partial X_{i,j}} f(X_i^{c|=z}).$$

□

Theorem 3.2 (Derivative of $L_1^{PDP_{centred},\,r}$ (3.9)). Let $f : \mathbb{R}^{nxp} \longrightarrow \mathbb{R}$ represents the differentible function that is explained by PDP. Let Z be the set of points that are used to calculate PDP. Then

$$\nabla_{X_{-c}} L_1^{PDP_{centred},r}(X) = \frac{2}{n|Z|} \sum_{z \in Z} \left(\nabla_{X_{-c}} f(X^{c|=z}) - \frac{\sum_{z' \in Z} \nabla_{X_{-c}} f(X^{c|=z'})}{|Z|} \right)$$

$$\cdot \left(\left(PDP_c(X, z) - \frac{\sum_{z' \in Z} PDP_c(X, z')}{|Z|} \right) - \left(PDP_c(X', z) - \frac{\sum_{z' \in Z} PDP_c(X', z')}{|Z|} \right) \right).$$

Proof. We calculate the derivative with respect to a particular value $X_{i,j}$ in the dataset. Please note, that we want to leave column c intact, so $j \neq c$. We have

$$\frac{\partial L_1^{PDP_{centred},\,r}(X)}{\partial X_{i,j}} =$$

$$\frac{\partial}{\partial X_{i,j}} \frac{1}{|Z|} \sum_{z \in Z} \left(\left(PDP_c(X, z) - \frac{\sum_{z' \in Z} PDP_c(X, z')}{|Z|} \right) - \left(PDP_c(X', z) - \frac{\sum_{z' \in Z} PDP_c(X', z')}{|Z|} \right) \right)^2$$

$$= \frac{2}{|Z|} \sum_{z \in Z} \left(\left(PDP_c(X, z) - \frac{\sum_{z' \in Z} PDP_c(X, z')}{|Z|} \right) - \left(PDP_c(X', z) - \frac{\sum_{z' \in Z} PDP_c(X', z')}{|Z|} \right) \right)$$
$$\cdot \left(\frac{1}{n} \sum_{k=1}^{n} \frac{\partial}{\partial X_{i,j}} f(X_k^{c|=z}) - \frac{1}{|Z|} \sum_{z' \in Z} \frac{1}{n} \sum_{k=1}^{n} \frac{\partial}{\partial X_{i,j}} f(X_k^{c|=z}) \right)$$
$$= \frac{2}{n|Z|} \sum_{z \in Z} \left(\left(PDP_c(X, z) - \frac{\sum_{z' \in Z} PDP_c(X, z')}{|Z|} \right) - \left(PDP_c(X', z) - \frac{\sum_{z' \in Z} PDP_c(X', z')}{|Z|} \right) \right)$$
$$\cdot \left(\frac{\partial}{\partial X_{i,j}} f(X_i^{c|=z}) - \frac{1}{|Z|} \sum_{z' \in Z} \frac{\partial}{\partial X_{i,j}} f(X_i^{c|=z}) \right).$$

□

Theorem 3.3 (Derivative of $L_1^{ALE,\,t}$ and $L_1^{ALE,\,r}$ (3.8)). Let $f : \mathbb{R}^{nxp} \longrightarrow \mathbb{R}$ represents the differentible function that is explained by ALE. Let Z be the set of points that are used to calculate ALE. Let $T : Z \to \mathbb{R}$. Finally, let $X' \in \mathbb{R}^{nxp}$ be the original dataset. Then

$$\nabla_{X_{-c}} L_1^{ALE,\,t}(X) = \frac{2}{|Z|} \sum_{z \in Z} \left(ALE_c(X, z) - T(z) \right) \nabla_{X_{-c}} ALE_c(X, z),$$

$$\nabla_{X_{-c}} L_1^{ALE,\,r}(X) = \frac{2}{|Z|} \sum_{z \in Z} \left(ALE_c(X, z) - ALE_c(X', z) \right) \nabla_{X_{-c}} ALE_c(X, z).$$

Proof. This equation comes from application of chain rule. Derivation is very similar to Proof of Theorem 3.2. □

Theorem 3.3 requires calculated gradient of the ALE method, which is presented in Lemma 3.4.

Lemma 3.4. Let $f : \mathbb{R}^{nxp} \longrightarrow \mathbb{R}$ represents the differentible function that is explained by ALE. Let $Z = (z_1, ..., z_g)$ be the set of points that are used to calculate ALE. Then

$$\frac{\partial ALE_c(z_K)}{\partial X_{i,j}} = \sum_{k=1}^{K} \left[\frac{1}{\sum_{l=1}^{n} w_l(z_k)} w_i(z_k) \left(\frac{\partial f}{\partial X_{i,j}} \left(X_i^{c|=z_k} \right) - \frac{\partial f}{\partial X_{i,j}} \left(X_i^{c|=z_{k-1}} \right) \right) \right].$$

Proof. We drop the most inner $\sum$ sign because only one of the terms contains variable j – variable that is used for differentation. □

Corollary 3.5 shows that the derivative of the next point z_{K+1} can be calculated from the derivative of z_K. This observation is particularly useful in the implementation of the algorithm; simplifies, and speeds up the calculation.

Corollary 3.5. Note that

$$\frac{\partial ALE_c(z_{K+1})}{\partial X_{i,j}} = \frac{\partial ALE_c(z_K)}{\partial X_{i,j}} + \frac{1}{\sum_{l=1}^{n} w_l(z_k)} w_i(z_{K+1}) \left(\frac{\partial f}{\partial X_{i,j}} \left(X_i^{c|=z_{K+1}} \right) - \frac{\partial f}{\partial X_{i,j}} \left(X_i^{c|=z_K} \right) \right).$$

Theorem 3.6 (Derivative of L_2). If $X \in \mathbb{R}^{nxp}$, then

$$\nabla_{X_{-c}} L_2(X) = \frac{2}{np}(X - X').$$

Proof. Note that

$$\frac{\partial L_2(X)}{\partial X_{i,j}} = \frac{\partial}{\partial X_{i,j}} \sum_{k,l} (X_{k,l} - X'_{k,l})^2 = \frac{\partial}{\partial X_{i,j}} (X_{i,j} - X'_{i,j})^2 = \frac{\partial}{\partial X_{i,j}} 2(X_{i,j} - X'_{i,j}).$$

□

Theorem 3.7 (Derivative of L_3). Let $f : \mathbb{R}^{nxp} \longrightarrow \mathbb{R}$ represents the differentible function. Then

$$\nabla_{X_{-c}} L_3(X) = \frac{2}{n} \nabla_{X_{-c}} f(X) \circ \left(f(X) - f(X') \right),$$

where $\circ$ denotes an element-wise product. Right vector is assumed to be vertical and multiplication is performed for each column.

Proof. Note that

$$\frac{\partial L_3(X)}{\partial X_{i,j}} = \frac{\partial}{\partial X_{i,j}} \frac{1}{n} \sum_{k=1}^{n} \left(f(X_i - f(X'_i) \right)^2 = \frac{2}{n} \left(f(X_i) - f(X'_i) \right) \frac{\partial}{\partial X_{i,j}} f(X_i).$$

□

We present the general schema of the robustness check for PDP explanation in Algorithm 7, and the schema of the targeted attack on ALE in Algorithm 8. We implemented the remaining considered attacks accordingly.

Assumption: The main practical assumptions of the attack are that the input data is standardized, and the model provides an interface for differentiation of output in respect to the input; thus, we apply it to the neural networks.

Standarization of the dataset is mostly required in the second term of the loss function.

Note: It is very easy to set some of the columns in the dataset or even particular values as constants. It means that during the attack they are not considered as arguments of loss function; thus, they do not change. It is done simply by setting to 0 partial derivatives corresponding to

chosen values in gradient of the loss function.

ALGORITHM 7: Gradient-based robustness check.

Input: f, X', g, c, C, α, β
Output: $g_c(X)$, X

$\hat{y} \leftarrow f(X')$
$X \leftarrow X' + noise()$
while *iteration* $<$ *max_iterations* ***or*** *relative_change* $< \epsilon$ **do**
  $output \leftarrow f(X)$
  $\nabla input \leftarrow gradient(f,\ X,\ output)$
  `// Keep side results of explanation calculations to use them in further gradient calculations`
  $result\_explanation,\ side\_results \leftarrow g_c(X)$
  $l1 \leftarrow \nabla L_1^{g,\ r}(f,\ result\_explanation,\ side\_results,\ g_c(X'))$
  $l2 \leftarrow \nabla L_2(X,\ X')$
  $l3 \leftarrow \nabla L_3(\nabla input,\ output,\ \hat{y})$
  $\nabla L \leftarrow l1 + \alpha \cdot l2 + \beta \cdot l3$
  $\nabla L \leftarrow set\_to\_0(\nabla L,\ C)$
  $X \leftarrow X - \eta \cdot adam(\nabla L)$
end

ALGORITHM 8: Gradient-based targeted attack.

Input: f, X', g, c, T, C, α, β
Output: $g_c(X)$, X

$\hat{y} \leftarrow f(X')$
$X \leftarrow X' + noise()$
while *iteration* $<$ *max_iterations* ***or*** *relative_change* $< \epsilon$ **do**
  $output \leftarrow f(X)$
  $\nabla input \leftarrow gradient(f,\ X,\ output)$
  `// Keep side results of explanation calculations to use them in further gradient calculations`
  $result\_explanation,\ side\_results \leftarrow g_c(X)$
  $l1 \leftarrow \nabla L_1^{g,\ t}(f,\ result\_explanation,\ side\_results,\ T)$
  $l2 \leftarrow \nabla L_2(X,\ X')$
  $l3 \leftarrow \nabla L_3(\nabla input,\ output,\ \hat{y})$
  $\nabla L \leftarrow l1 + \alpha \cdot l2 + \beta \cdot l3$
  $\nabla L \leftarrow set\_to\_0(\nabla L,\ C)$
  $X \leftarrow X - \eta \cdot adam(\nabla L)$
end

3.3.1. Parameters

Learning rate, denoted as η, controls a step size in each iteration. Greater learning rate usually results in faster algorithm convergence but may lead to worse solutions, wherein the worst-case scenario, the algorithm might not converge at all. On the other hand, a lower learning rate may result in too slow convergence and not optimal result. We are using by default 0.01 as learning rate for standardized data. Note that for not standardized data, this value can be of any magnitude.

Epsilon, denoted as ϵ, is used in early stopping. If the relative change of the loss function between the iterations is lesser than epsilon, the algorithm stops because it is very probable that there will be no much improvement in the next iterations.

4. Implementation in Python

The developed algorithms are wrapped in the Python package named **attackpdp**. In general, we have one class per algorithm, and in each one method per attack strategy. The constructor of each class takes as an argument the `dalex.Explainer` object and required parameters, while methods consists of the parameters that would be considered as variables in algorithm fitting. First class implements the genetic-based algorithm for attacks on explanations of black-box models, while the second class implements the gradient-based algorithm for attacks on explanations of neural networks.

4.1. Technology selection

The main Python library, which we use in this implementation is the `numpy` package for scientific computing [Harris et al., 2020]. It provides an interface for n-dimensional arrays and numerical computing tools, which are the baseline of our algorithms. It is also compatible with all of the other package dependencies. The next technology used in this project is the `tensorflow` package (in version `v2` or higher) – a framework for neural networks [Abadi et al., 2016]. We mainly rely on its functionalities in the gradient-based attack, which is only for neural networks; in our implementation, only for `tensorflow` neural network models. On the other hand, the genetic-based algorithm in theory can be applied to any model, but in practice we aim at complying with the most popular machine learning frameworks (e.g. `scikit-learn` [Pedregosa et al., 2011], `lightgbm` [Ke et al., 2017]). To achieve that, we utilize the `dalex` package [Baniecki et al., 2020], which implements model explanations for machine learning predictive models. Additionally, it aims at the unification of various models and packages for machine learning, which is crucial in practice. Finally, we utilize the `pandas` package for dataframe structures [Wes McKinney, 2010].

4.2. Package description

The `attackpdp` Python package is a first to provide easily accessible attacks for model explanations due the compatibility with the `dalex` package It implements methods described in this work, in all of their versions and parametrizations. It also allows to reproduce the results presented in Chapter 5.

We divided the package into three modules:

- `attacks` – main module containing classes that implement the algorithms,
- `datasets` – loads examplary datasets used in examples and experiments,
- `utils` – provides utility functionalities, e.g. implementation of the Adam optimizer.

The `attack` module is further divided into three parts, where the main file contains the abstract class, and then `gradient_attack` and `genetic_attack` submodules contain the main classes. This gives a level of abstraction to this package that helps the development and provides a high-level representation. The main `Attack` class is an abstract class that defines the common parameters (Table 4.1) and methods (Table 4.2) for both algorithms.

Table 4.1: Common parameters for the `Attack` class.

Name	**Type**	**Description**
`explainer`	`dalex.Explainer`	object containing model and data
`variable`	`str`	variable for explanation (c)
`explanation_type`	`'pdp'` or `'ale'`	attacked explanation
`constant`	list of `str`	variables that remain unchanged during the attack (e.g. categorical variables)
`n_grid_points`	`int`	number of points in which PDP or ALE are calculated (Z), used only if `splits` parameter was not explicitly defined (see Tables 4.3 and 4.4)

Both, genetic-based and gradient-based algorithm do not differ in abstract sense at all, they do differ in the implementation of the optimization. `GradientAttack` and `GeneticAttack` classes extend the `Attack` class, implementing `robustness_check()` and `targeted_attack()` methods for performing the attacks, which have common parameters presented in Tables 4.3 and 4.4.

Table 4.2: Common methods for the `Attack` class.

Name	Description
`robustness_check()`	implementation of the robustness check strategy
`targeted_attack()`	implementation of the targeted attack strategy
`plot_losses()`	visualise fitting of the algorithm

Table 4.3: Common parameters for the `robustness_check()` method.

Name	Type	Description
`splits`	`np.ndarray`	list of `variable` values (Z), by default it is uniformly distributed over the min-max range
`alpha`	`float`	second term regularization parameter, the higher α, the higher impact L_2 has on a loss function
`beta`	`float`	third term regularization parameter, the higher β, the higher impact L_3 has on a loss function
`max_iter`	`int`	maximum number of algorithm iterations

Table 4.4: Common parameters for the `targeted_attack()` method.

Name	Type	Description
`target`	function or `np.ndarray`	either a function that given `splits` returns the targeted explanation, or a list of explanation values corresponding to `splits`
`splits`	`np.ndarray`	list of `variable` values (Z), by default it is uniformly distributed over the min-max range
`alpha`	`float`	second term regularization parameter, the higher α, the higher impact L_2 has on a loss function
`beta`	`float`	third term regularization parameter, the higher β, the higher impact L_3 has on a loss function
`max_iter`	`int`	maximum number of algorithm iterations

4.3. Implementation details

Available to the users are `GeneticAttack` and `GradientAttack` classes. We recognize further package elements for the development purposes.

4.3.1. Genetic-based attack

The `GeneticAttack` class implemented in the `attacks.genetic_attack` submodule introduces attack-specific parameters and behaviour. Its methods use the implementation of explanations and loss functions included in the `explanations` and `loss` submodules respectively. The main implementation of the class, with its attacks, is included in the `object.py` file because of the explanation-agnostic nature of the algorithm.

4.3.2. Gradient-based attack

Respectively, the `GradientAttack` class implemented in the `attacks.gradient_attack` submodule introduces attack-specific parameters and behaviour. Its methods use raw algorithm implementations included in the `implementation` submodule. Most of the implementation is included in the `gradient_attack_abstraction.py` file, because of various similarities between the attacks on PDP and ALE. Nevertheless, the `gradient_attack_pdp.py` and `gradient_attack_ale.py` files contain explanation-specific parts of the algorithm, most notably the first term derivative.

4.4. Example of usage

We present a basic example usage of the `attackpdp` package. For simplicity, we use the `heart1` dataset included in the `attackpdp` package, which is further described in Section 5.1. We want categorical variables in the data to remain constant during the attack; thus, we keep their names in `constant_variables`. In this case, we know that the categorical variables have less than six unique values.

```
# import the package
import attackpdp
# load the heart1 dataset
data = attackpdp.datasets.load_heart1()
# split the dataset into variables and the target
X, y = data.drop(columns="target"), data.target
# save names of the categorical variables
constant_variables = X.columns[X.nunique() < 6]
```

4.4.1. Genetic-based robustness check for PDP

Genetic-based attack is model-agnostic, which means that we can attack explanations of any predictive model (provided the predict interface). In this example we use a `SVC` model from the `scikit-learn` package. `attackpdp` is designed to work with `dalex`; thus, after model fitting, we need to wrap it with the `dalex.Explainer` object.

```
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
# make a pipeline SVC model which requires scaling
model = Pipeline([
    ('scaler', StandardScaler()),
    ('estimator', SVC(C=10, probability=True))
])
model.fit(X, y)

import dalex as dx
# create an Explainer
explainer = dx.Explainer(model, X, y)
```

Next, we perform the robustness check for PDP explanation. We create the `GeneticAttack` object and use the `robustness_check()` method.

```
# initialize the GeneticAttack
genetic_attack = attackpdp.attacks.GeneticAttack(
    explainer,
    variable='age',
    explanation_type='pdp',
    constant=constant_variables,
    mutation_with_constraints=True
)
# perform the check
genetic_attack.robustness_check()
```

After the attack, several results and plotting options are available.

```
# plot losses
genetic_attack.plot_losses()

# plot the changed explanation
genetic_attack.plot_result_explanation()

# plot the changed data
genetic_attack.plot_result_data(constant=False)
```

4.4.2. Gradient-based targeted attack on ALE

Gradient-based attack is model-specific, specifically it requires to compute gradients of the model's output. Thus, we implemented this type of attack only for `tensorflow` models. It case of this implementation, it is important to scale the data inbefore the process.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.losses import BinaryCrossentropy
import numpy as np
# scale data
columns = X.columns
X_std = (X.values - np.mean(X.values, axis=0)) / X.values.std(axis=0)
# make a Sequential model with dense layers
model = Sequential([
  Dense(16, input_dim=X.shape[1], activation='relu'),
  Dense(16, activation='relu'),
  Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss=BinaryCrossentropy(), metrics=['acc'])
model.fit(X_std, y.values)

import dalex as dx
import pandas as pd
# create an explainer
explainer = dx.Explainer(model, pd.DataFrame(X_std, columns=columns), y)
```

We perform the targeted attack on ALE explanation. We create the `GradientAttack` object, and define the targeted function; then use the `targeted_attack()` method.

```
# define the target
target = lambda x: 0.05*x + 0.1
# initialize the GradientAttack
gradient_attack = attackpdp.attacks.GradientAttack(
    explainer,
    variable='age',
    explanation_type='ale',
    constant=constant_variables
)
# perform the targeted attack
gradient_attack.targeted_attack(target=target, max_iter=100)
# plot losses
gradient_attack.plot_losses()
```

5. Experiments

In this chapter we evaluate the implemented attacks and answer key questions, e.g. how model complexity correlates with a successful attack.

5.1. Experimental setting

We first define all of the datasets and models used to perform the experiments.

5.1.1. Datasets

- `friedman1` available through `attackpdp.datasets.load_friedman1()`.

  "Friedman 1" regression problem [Friedman, 1991; Breiman, 1996] generated with `sklearn.datasets.make_friedman1(n_samples=1000,n_features=10,random_state=0)`. Inputs X are independent variables uniformly distributed on the interval $[0, 1]$, while the target y is created according to the formula:

  $$y(X) = 10 \cdot \sin(\pi \cdot X_0 \cdot X_1) + 20 \cdot (X_2 - 0.5)^2 + 10 \cdot X_3 + 5 \cdot X_4.$$

  Only 5 variables are actually used to compute y, while the remaining are independent of.

- `friedman2` available through `attackpdp.datasets.load_friedman2()`.

  "Friedman 2" regression problem [Friedman, 1991; Breiman, 1996] generated with `sklearn.datasets.make_friedman2(n_samples=1000,random_state=0)`. Inputs X are 4 independent variables uniformly distributed on the intervals:

  $$0 \leq X_0 \leq 100, \quad 40\pi \leq X_1 \leq 560\pi,$$

  $$0 \leq X_2 \leq 1, \quad 1 \leq X_3 \leq 11.$$

The target y is created according to the formula:

$$y(X) = (X_0^2 + (X_1 \cdot X_2 - 1/(X_1 \cdot X_3))^2)^{0.5}.$$

- `heart1` available through `attackpdp.datasets.load_heart1()`. "Heart 1" classification problem from Kaggle[1]. There are 303 observations, 5 continuous variables, 8 discrete variables, and an evenly-distributed binary target.
- `heart2` available through `attackpdp.datasets.load_heart2()`. "Heart 2" classification problem is the "Heart 1" problem without discrete variables, which simplifies the predictive task and makes it more challenging.

5.1.2. Experiments setup

For the task on `heart1` dataset, we set 8 categorical variables as constant during the performed attack algorithms. This is because we mainly rely on incremental change in the values of continuous variables, and categorical variables are out of scope of this work.

Furthermore, in all the experiments α and β parameters are set to 0; thus, an algorithm has no limits in changing the data or model's predictions. This is to perform a *sanity check*, which shows the greatest possible change in model explanations without any restrictions. Such strategy could always be the first step in the attack; if the explanation change without any restrictions is not noticeable, then there is no sense in tuning the method.

Finally, for the `heart` predictive tasks we target explanations of the `age` variable, while for the `friedman` predictive tasks we target the first variable `0`.

5.2. Genetic-based attack

We start evaluating the genetic-based attack by showcasing a simple result of the targeted attack on PDP method in Figure 5.1. For the support vector machine model and the `age` explanatory variable, two targets are possible: increasing or decreasing monotonicity. Data poisoning results in a drastic change of the visual explanation, which will be an increasing security concern throughout the remaining part of work. In further experiments, we measure $L_1^{g,\,r}$ distance between explanations (named `loss`) to identify the power of robustness check. **Important:** There is no possibility to compare loss values between the datasets/tasks, since the magnitude of these values depends on scaling.

[1]Source of the dataset can be foud at `https://www.kaggle.com/ronitf/heart-disease-uci`.

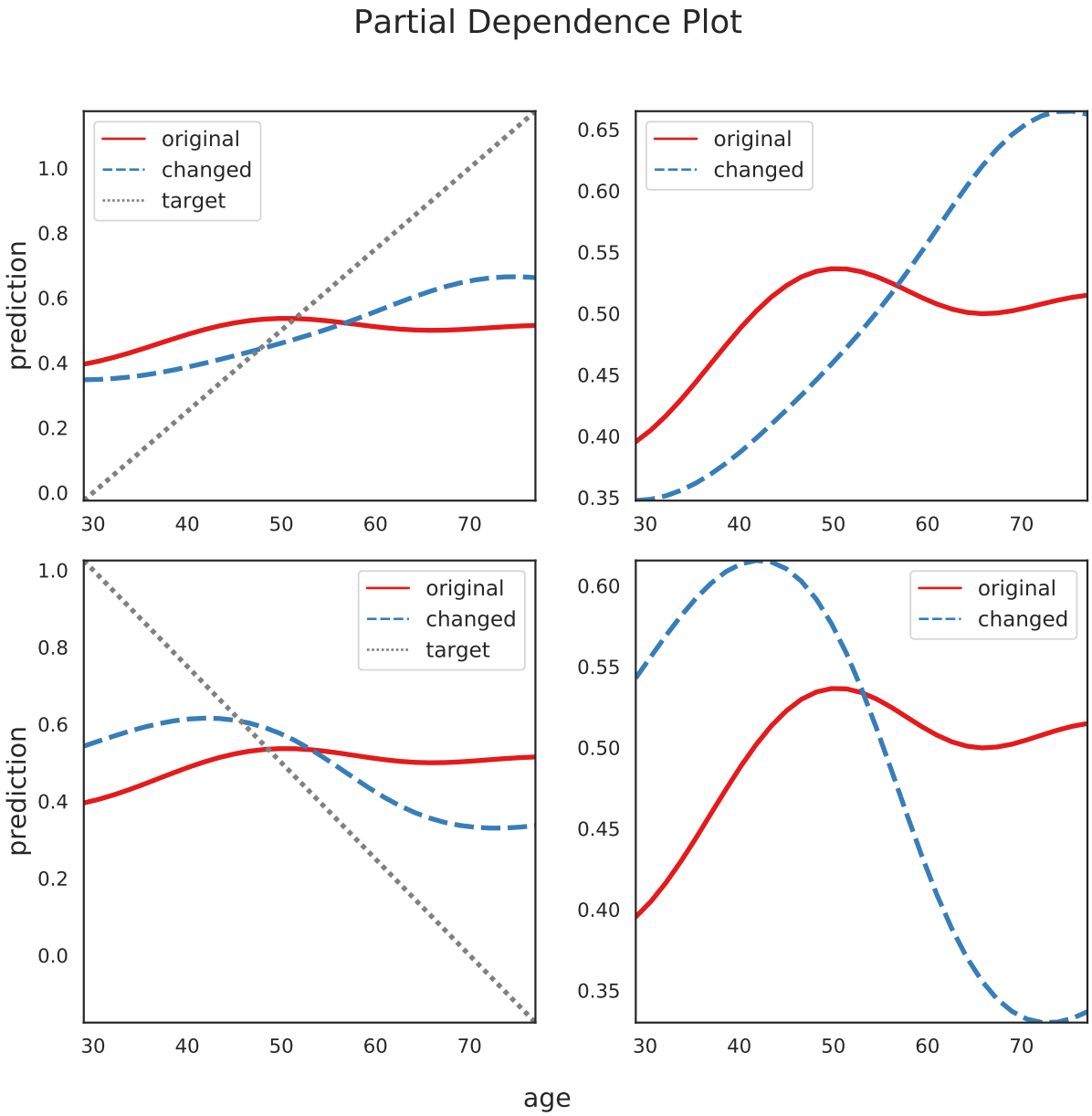

Figure 5.1: Genetic-based attacks on PDP of the SVM model trained on `heart2` dataset. The original visual explanation (red line) oscilates on the decision boundary of `0.5`. Setting two different targets for the attacks (grey lines) produces contradictory results (blue lines) which we call a security breach of Explainable AI methods.

Explanations of which machine learning models are vulnerable?

Four introduced datasets were split in 4 : 1 (train : test) ratio. Models were evaluated on the test set, and explanations were computed on the train set. For `heart` classification tasks we measure AUC, while for `friedman` regression tasks we measure R-squared. All the considered models come from the `scikit-learn` package [Pedregosa et al., 2011], with exception of gradient boosting machine from the `lightgbm` package [Ke et al., 2017], as it is more efficient. These were trained mostly with default parameters as we consider predictive tasks to be easy.

We use: `LinearRegression` & `LogisticRegression` (LM), `DecisionTreeRegressor` & `DecisionTreeClassifier` (DT), `RandomForestRegressor` & `RandomForestClassifier` (RF), `LGBMRegressor` & `LGBMClassifier` (GBM), `SVR(C=100)` & `SVC(C=10, tol=1)` (SVM), `KNeighborsRegressor(n_neighbors=5)` & `KNeighborsClassifier(n_neighbors=5)` (KNN), `MLPRegressor(max_iter=4000,hidden_layer_sizes=(32, 32))` & `MLPClassifier(max_iter=3000, hidden_layer_sizes=(32, 32))` (NN).

Figure 5.2 shows results for the robustness checks for PDP and ALE methods by various machine learning models (columns) on all datasets (rows). Additionally, we see the impact of adding the constraints to the optimization algorithm, which requires changing data only within the original variable range. Each point corresponds to one algorithm evaluation, and the loss equals to $L_1^{g,\,r}$; thus, in the robustness check strategy, these are negative values that we aim to minimize.

The first observation is that basic linear models are indifferent to the attack, which is expected, as their simple nature allows us to interpret variable coefficients. Further, tree-based models, random forest and gradient boosting machine, at first sight, prove robustness against the attacks on PDP and ALE; unlike the explanation of simple decision tree, which on several occasions was fooled. Secondly, the worst candidate for an explanation with PDP or ALE is the support vector machine. Practically in any experimental setting, it is easy to change the monotonicity of visual explanations for this model. This result may be treated as a warning for machine learning practitioners wanting to use such a model in a scenario where data shift is possible.

Looking at the parametrization of genetic-based algorithm mutation, we can see that when data change is constrained to the range of variables, an attack's outcomes have practically no variance. This phenomenon may be due to easier and faster algorithm convergence. Removing the constraints gives varied results, which may be attributed to a larger solution space available.

The most straightforward predictive task is `friedman2`, where all models achieve ideal performance on the test set (denoted as additional red labels). Only the explanations of the k-nearest neighbours model are easy to manipulate, which in some sense may set it as the weakest model to trust. Then, comparing k-nearest neighbours and neural networks on other datasets showcases little correlation between model performance and a probability of successful attack.

Looking at the performance measures, we can observe that the decision tree model had lower predictive performance than ensemble methods, which is to be expected. However, it regularly entails lower loss values, so is the model complexity correlated with an attack performance? We answer this question in the next section.

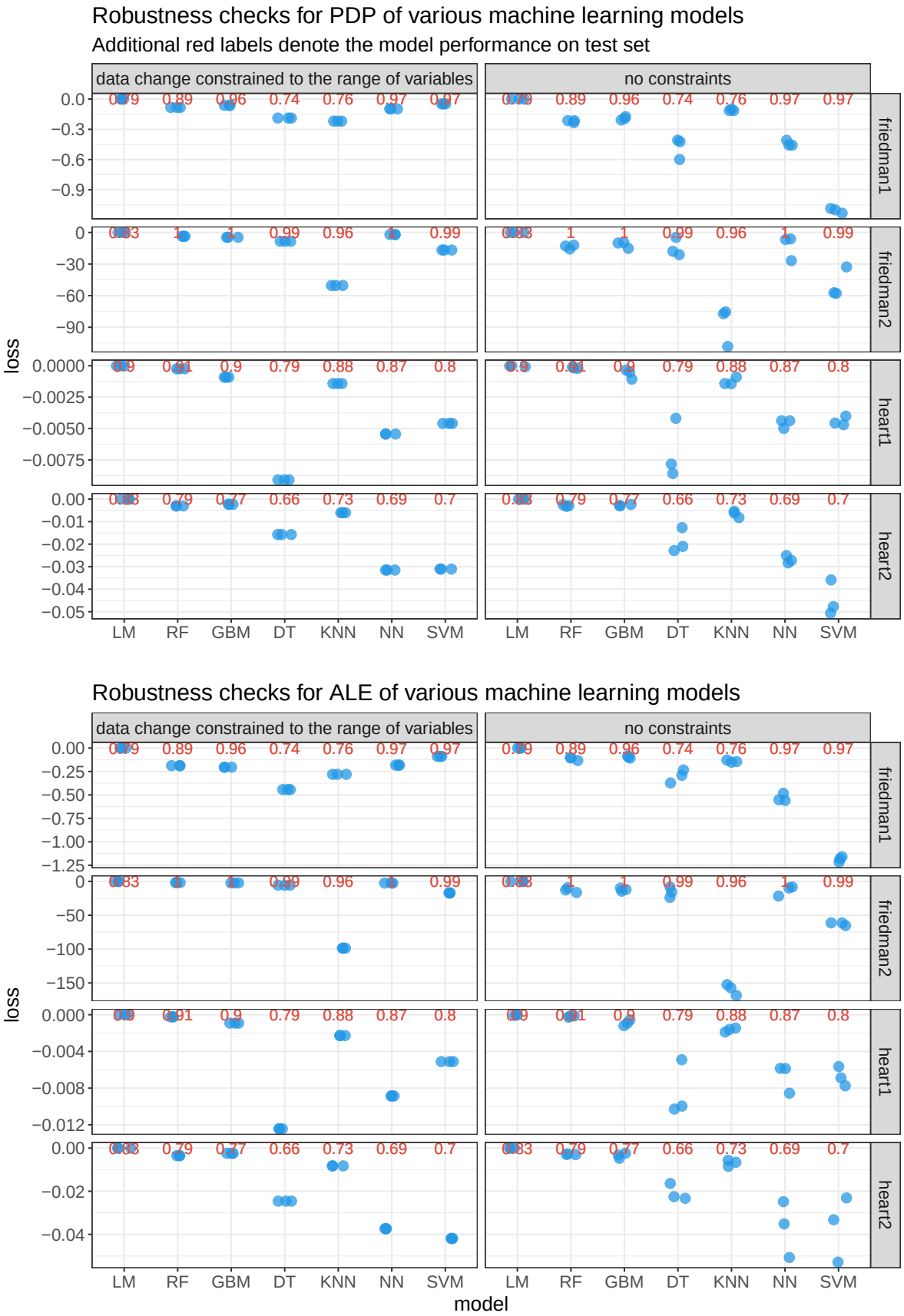

Figure 5.2: Robustness checks for PDP and ALE of various machine learning models on all four datasets. We can see that ensemble models provide robustness capabilities, while neural networks and other distance-based models seem to have vulnerable explanations.

Finally, comparing both plots in Figure 5.2 we cannot say that there are differences between the results of attacks on PDP or ALE.

How much model complexity corresponds with explanation vulnerability?

In this experiment, the datasets were split in 4 : 1 (train : test) ratio. Models were evaluated on the test set, and explanations were computed on the train set. For `heart` classification tasks we measure AUC, while for `friedman` regression tasks we measure R-squared. Random forest and multilayer perceptron come from the `scikit-learn` package, while gradient boosting machine comes from the `lgbm` package. These were trained mostly with default parameters.

We use: `RandomForestRegressor` & `RandomForestClassifier` with `n_estimators` from the set `[10, 20, 40, 80, 160, 320]`; `LGBMRegressor` & `LGBMClassifier` with `n_estimators` from the set `[10, 20, 40, 80, 160, 320]`; `MLPRegressor(max_iter=10000)` & `MLPClassifier(max_iter=10000)` with `hidden_layer_sizes` from the set `[(16,), (32,), (16, 16), (32, 32), (16, 32, 16), (32, 64, 32)]`.

Figure 5.3 shows the results of the robustness checks for PDP and ALE methods of gradient boosting machine and random forest models on all four datasets by models' complexity. Looking at the most demanding `heart2` task, we see a relationship between models' complexity and the attacks' performance, without the interference of models' performance, as these are almost the same across the number of trees. Looking at the least demanding `friedman2` task, we see that even the smallest number of trees provides a ground for explanation change; while the explanations of complex underfitted and overfitted gradient boosting machine models showcase the lowest losses. Interestingly on the `friedman1` dataset, there are the only noticeable differences between the result of attacks on PDP or ALE – for both tree-ensemble models there is a high variance in robustness checks for ALE and low variance in robustness checks for PDP.

Which explanations can be trusted?

Comparing the magnitudes of loss values presented in Figures 5.2 and 5.3 we see that the absolute values for tree-ensemble models are about ten times lower. Even when considering various complexity levels and predictive tasks, gradient boosting machines and random forest models prove to have robust PDP and ALE explanations.

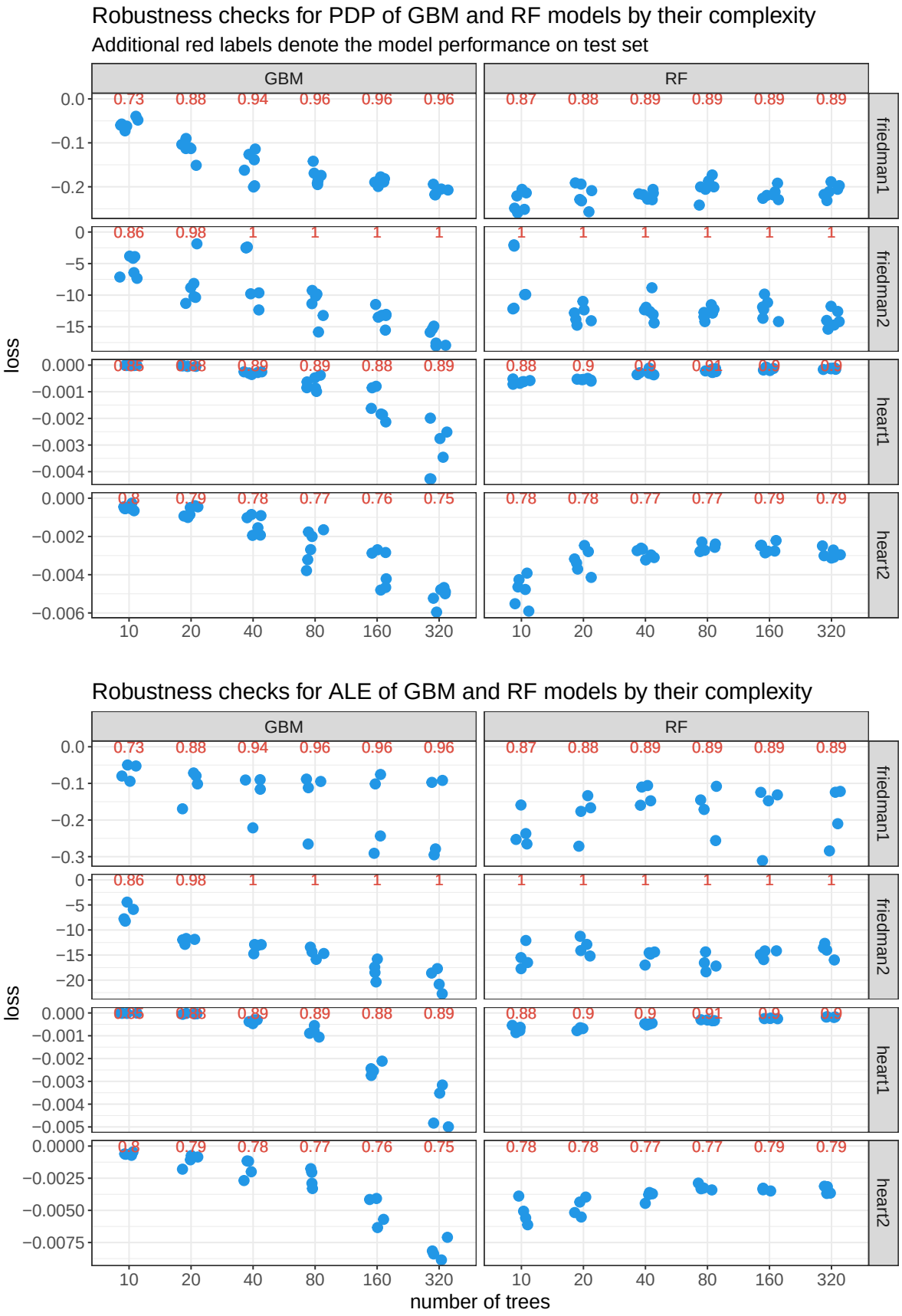

Figure 5.3: Robustness checks for PDP and ALE of gradient boosting machine and random forest models on all four datasets by models' complexity. We can see that more complex boosting models have vulnerable explanations, while the losses of attacks on random forest explanations are steady across various numbers of trees for all predictive tasks.

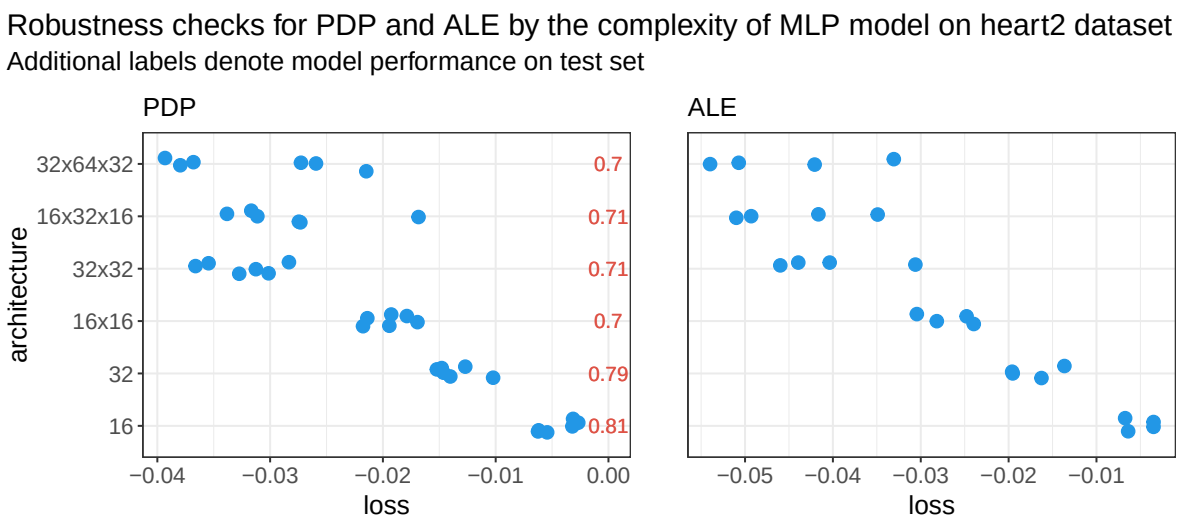

Figure 5.4: Robustness checks for PDP and ALE of multilayer perceptron model on `heart2` by models' complexity. We can observe that underfitted complex neural networks have vulnerable explanations.

Although the same experiment performed for multilayer perceptron model did not provide substantial valuable information, Figure 5.4 shows the most interesting result. We performed robustness checks for PDP and ALE on the `heart2` predictive task, where red labels present the AUC performance on the test set. Complex neural networks became underfitted on this task, which correlates with the attacks' performance. Both of these factors contribute to lower attacks' losses, which indicate vulnerable explanations. The next section presents in-depth results for neural network models obtained from experiments using the gradient-based attack.

5.3. Gradient-based attack

5.3.1. Experiments description

Experiments were performed with assumptions mentioned in Section 5.1.2. Additionally, for the gradient-based attacks, we created five versions of each dataset. These are constructed by adding noise variables – completely random variables which do not have any information about the target. There are five such versions with respectively: 0, 1, 2, 4 and 8 added noise variables. Dataset version with zero added variables is the original one. Variable `0` is the explained variable in `friedman1` while variable `age` is the explained variable in `heart1` dataset. We attacked both PDP and ALE explanations in our experiments. All versions of our methods were used; thus, a robustness check, centred robustness check, and targeted attack were performed for the PDP method; robustness check, and targeted attack for the ALE method. All attacks are

sanity checks – it is hard to compare restricted attacks because of different datasets' scales. Attacks were performed on nine neural networks with architectures: 8, 8x8, 8x8x8, 32, 32x32, 32x32x32, 128, 128x128 and 128x128x128. All activation functions are ReLU except output layers, where for `friedman1` we used linear activation and for `heart1` we used sigmoid. All categorical variables in the `heart1` dataset were left intact. Each attack was repeated 30 times with random initialization.

Experimental setup consists of:

1. two datasets: `friedman1` and `heart1`,
2. neural networks with hidden layers of sizes:
   (a) 8, 8x8, 8x8x8,
   (b) 32, 32x32, 32x32, 32x32x32,
   (c) 128, 128x128, 128x128x128,
3. ReLU activations in hidden layers,
4. robustness check and centred robustness check in case of the PDP,
5. robustness check for ALE,
6. targeted attacks on both PDP and ALE,
7. each task is repeated 30 times.

5.3.2. Results description

Figure 5.5 presents train and test metrics of all neural networks used in the experiments. The simplest models usually have much worse performance than the most complex ones. Models also span their performance relatively uniformly from the worst to the best. None of them is strongly overfitted, which means that this set of architectures cover models with very different performances and that fit correctly to the data. They can be all used in our experiments.

How much we can change PDP and ALE curve?

Figure 5.6 shows all 30 results of attacks on PDP explanation on `friedman1`'s first variable with neural network architecture 128x128x128. We can observe how much we can change the PDP curve and how each attack differs in its nature. The first subplot presents centred robustness check. All resulting curves have an average close to the original curve's average.

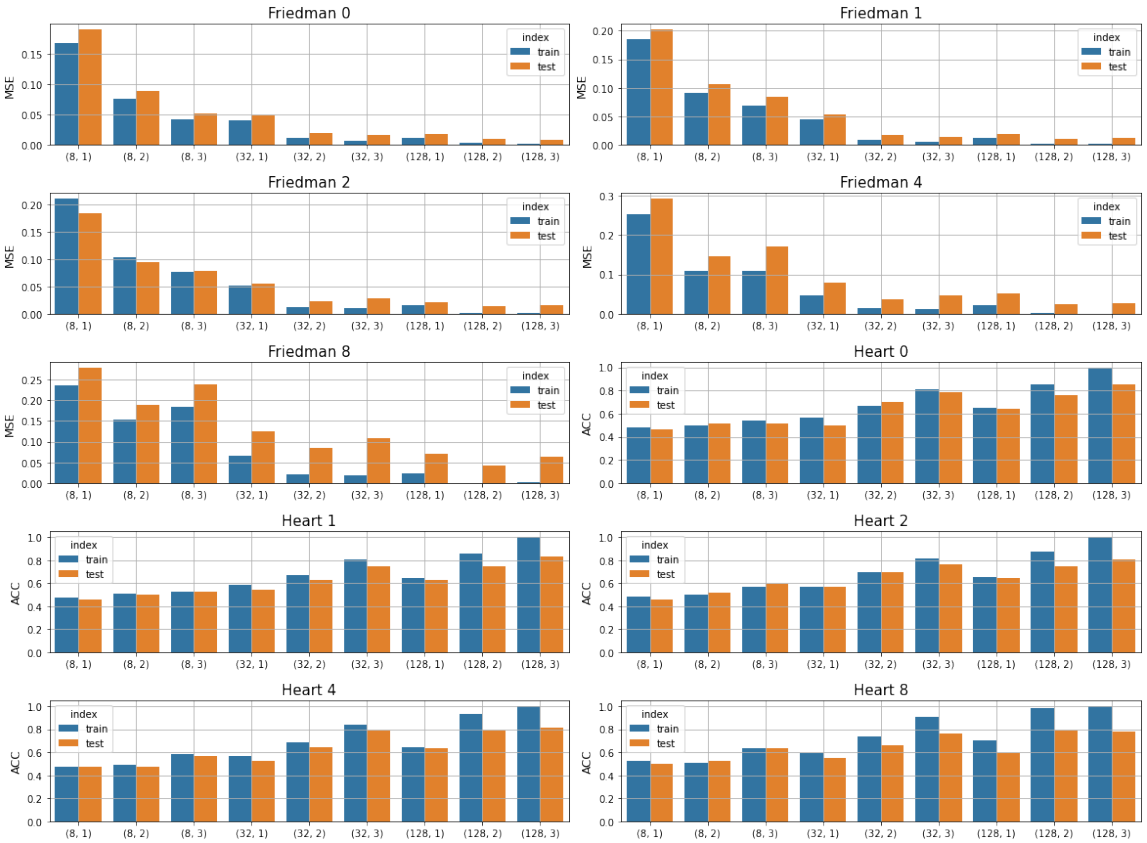

Figure 5.5: Performance of all neural networks used in experiments

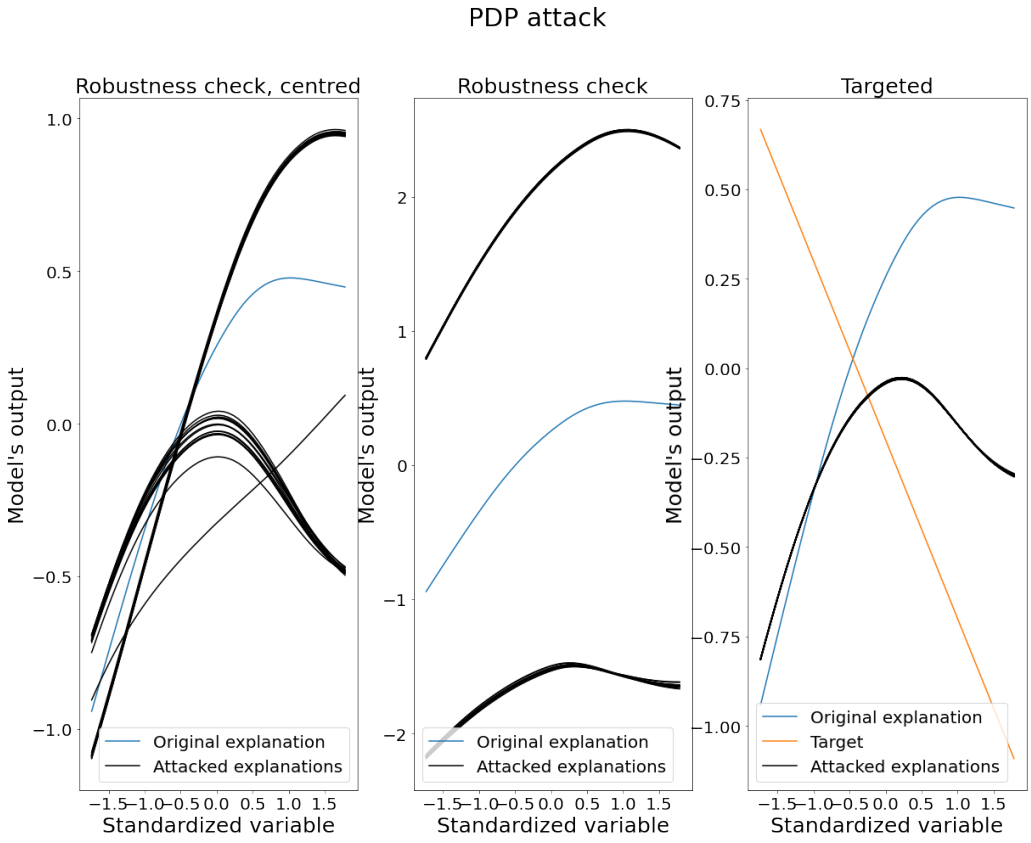

Figure 5.6: Comparison of PDP attacks on the `friedman1` dataset, 128x128x128 architecture

Thus the attack has to find changes in the curve that do not include shifting it upwards or downwards – changes that include changing the shape of the curve.

Two major solutions (different fooled explanations) were found, and one minor by our method. Major ones are bending the curve upward and downward, respectively. We will be able to observe such two solutions in robustness checks in all other examples. The minor solution consists of only one curve (only one attack iteration gave this result). It is worth emphasizing how different these two major solutions are. One shows that the model's predictions significantly increase with respect to the explained variable. Another shows that model's output increases and then decreases with respect to the explained variable, where maximum can be found around 0. Moreover, this is still the same model – only data has changed.

The second plot shows the results of a robustness check. This type of attack does not require maintaining the average value at the same level. Here, similarly as in the previous example, are two major solutions found by the attack. These are shifting upwards and downwards the whole curve without significant change in shape. Although shape has not changed significantly, the average prediction of the model strongly shifted. Please, note that we do not change the explained variable in our attacks. The whole change in prediction comes from other variables. These show how much we can change the model's responses with respect to some variable without actually changing it.

Third of attack types presented in Figure 5.6 is a targeted attack, where the orange line shows the target explanation. This type of attack resulted in only one primary solution – this type of behaviour is also observed in other targeted attacks examples. It is worth to note that the final explanations are not exactly the target line, but target forced explanations to change in some direction.

Figure 5.7 presents similar results but obtained for ALE explanation on `friedman1` dataset. ALE curve is hooked in 0; thus, we decided not to centre the manipulated curve. Left subplot shows robustness check while right subplot shows the targeted attack. Again, the robustness check resulted in two major solutions found by the gradient-based attack. One corresponds to bending the curve upward; the second one corresponds to bending the curve downward. Similarly to the results presented in Figure 5.6, two solutions strongly differ. One tells us that the model's output significantly increases with respect to the explained variable. The second one tells us that the model's response behaves parabolically – first increases then decreases.

Targeted attack in Figure 5.7 finds only one major solution – similarly to the PDP attack on `friedman1`. Figure 5.8, similarly to Figure 5.6, presents attacked PDP curves, but on `heart1` dataset. Gradient-based attack found two major solutions to centred robustness check, simi-

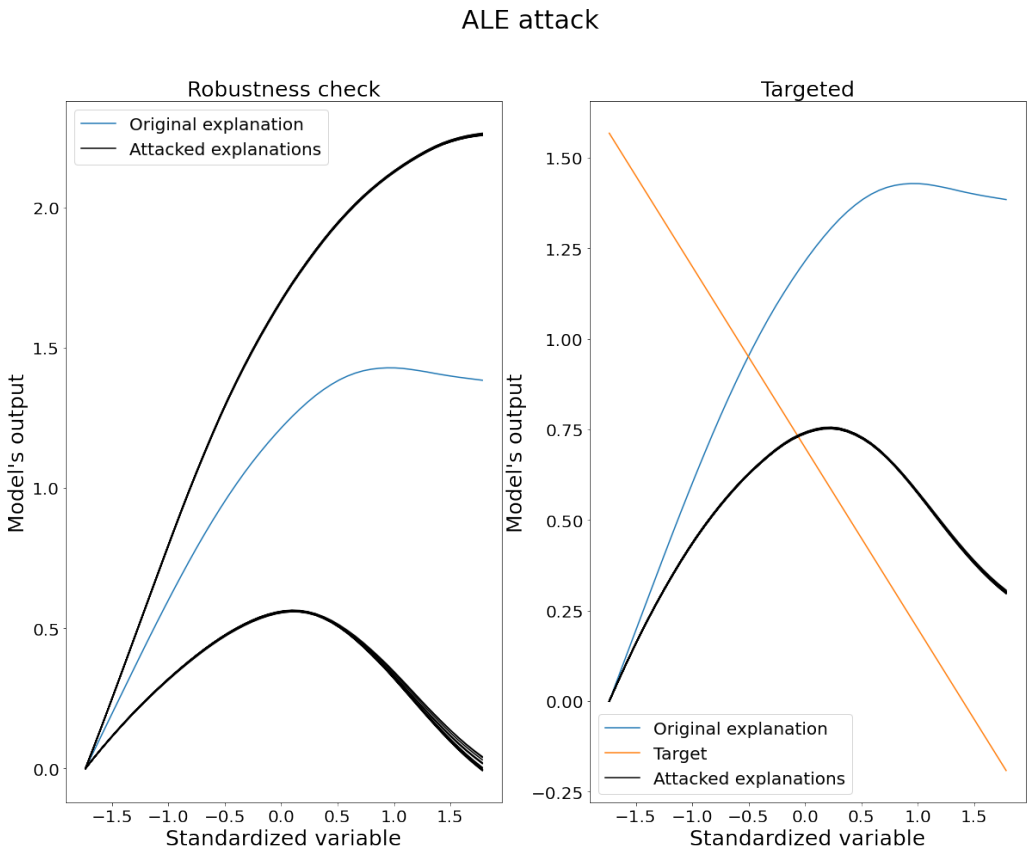

Figure 5.7: Comparison of ALE attacks on the `friedman1` dataset

larly to previous examples. Both suggest completely different conclusions about the model's behaviour. One of them tells us that the model's output strongly decreases with respect to explained variable `age`. This can be interpreted that the older people get, the higher probability of `heart1` disease they have. The second solution tells us a different story. It tells us that the higher age, the lower probability of becoming sick: two different interpretations, the same model. The original explanation can be interpreted as slightly increasing the probability of disease with respect to `age`.

Second subplot shows attacked explanations via robustness check. There are two major solutions, while some other minor do not seem to form any structure. Similarly to the middle subplot in Figure 5.6, they have the same shape to the original one, but they are just shifted upward and downward respectively. One may interpret both of these major solutions very differently. In the first scenario, there is a low probability of having a disease in general, while in the second scenario, there is a high probability.

The third subplot presents a targeted attack. We may observe again only one primary solution found be the method. This time the attacked explanation strongly resembles target function. Perhaps this is the outcome of the smaller dataset.

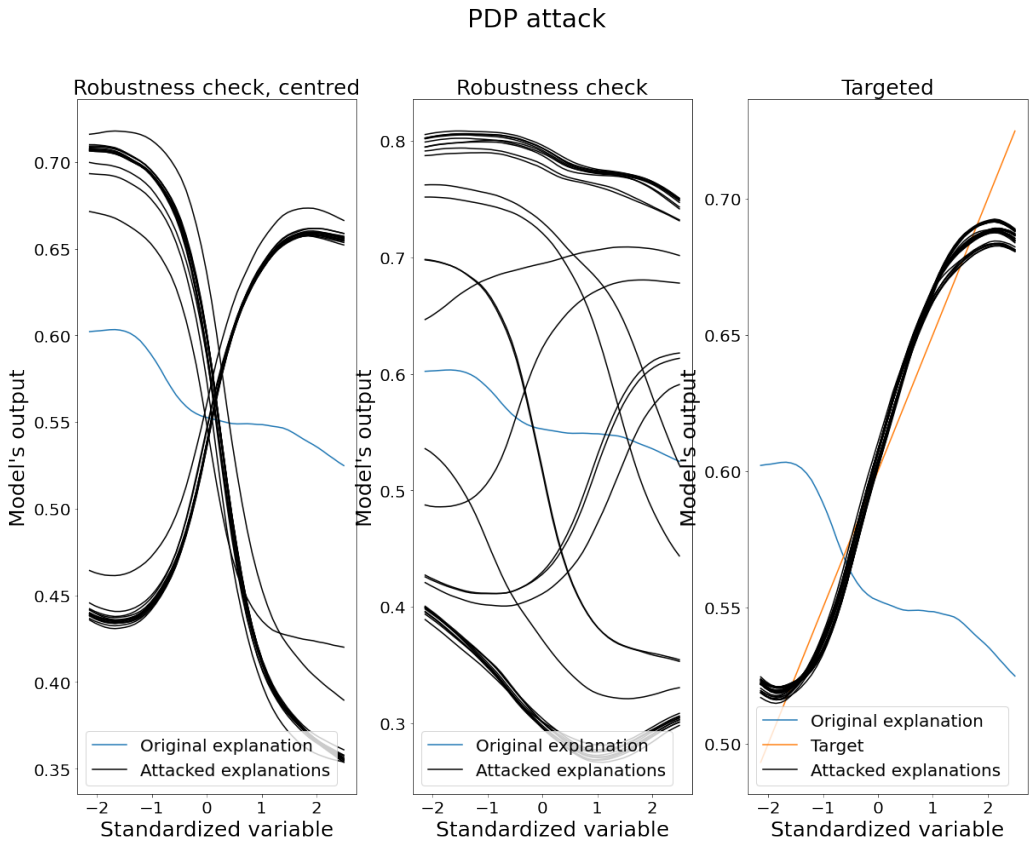

Figure 5.8: Comparison of PDP attacks on the `heart1` dataset.

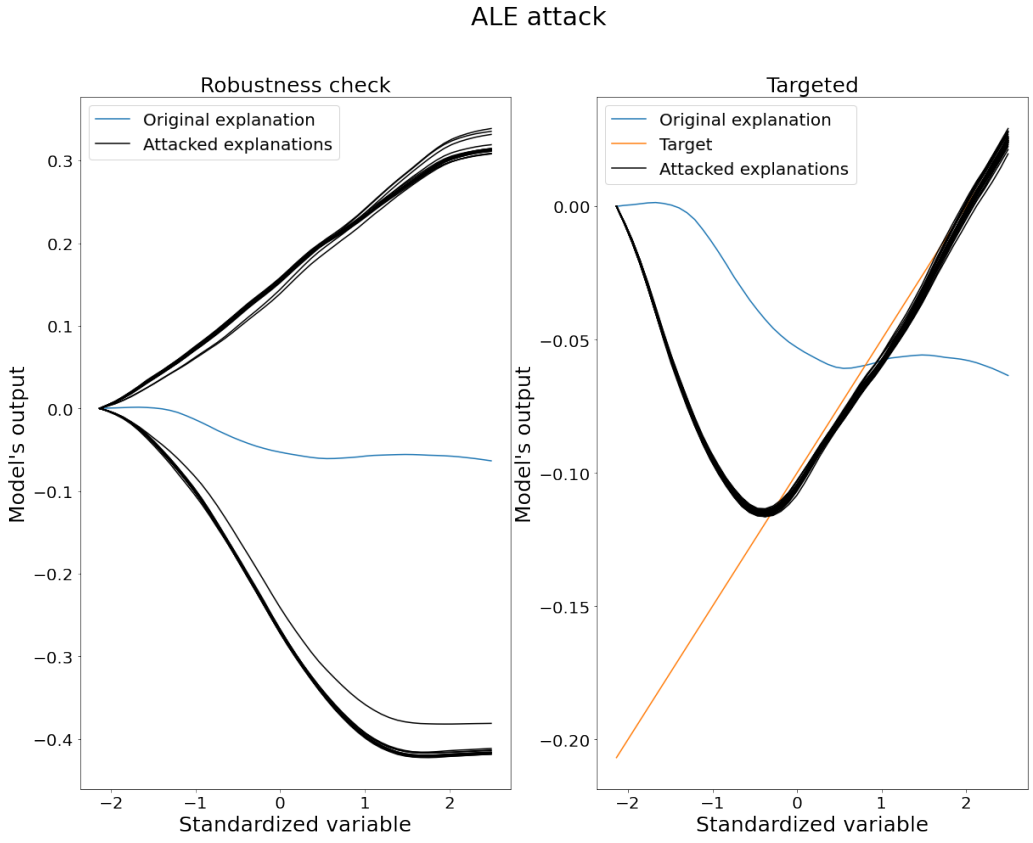

Figure 5.9: Comparison of ALE attacks on the `heart1` dataset.

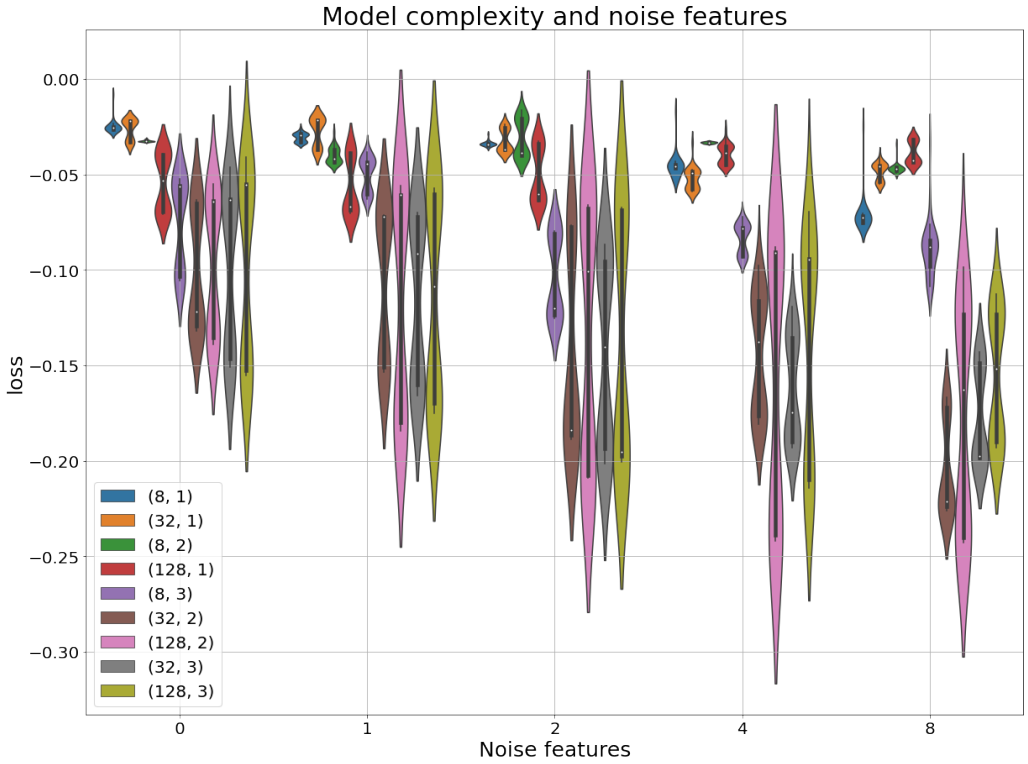

Figure 5.10: Comparison of PDP centred attacks' losses on different versions of `friedman1`.

Figure 5.9 shows result explanations after attacking ALE on `heart1` dataset. robustness check found two major solutions that strongly differ. One may tell two different stories depending on the chosen explanation. The targeted attack has only one solution. ALE method is hooked in 0, but the rest of the curve again very strongly resembles target function. Interpretation of such attacked explanation is very different from the original one's.

It may be a good idea to run robustness check multiple times to find the best solution – in terms of loss function or the general shape of the curve. The targeted attack does not seem to require repeating, which can be concluded from all the previous examples.

How much model complexity corresponds with vulnerability?

Figure 5.10 presents the distribution of final losses of centred robustness check for PDP on `friedman1` dataset grouped by the number of added noise variables for different neural network architectures. Neural network architectures are sorted by the number of parameters in each group.

Looking at Figures 5.10 and 5.11, the more added noise variables, the more significant changes in the curve we may expect. Although the model fits very well to the data, the usage of noise variables seems to be strong enough to push and bend the PDP curve. The loss seems to be almost linearly negatively correlated with the logarithm of added noise variables. Second obser-

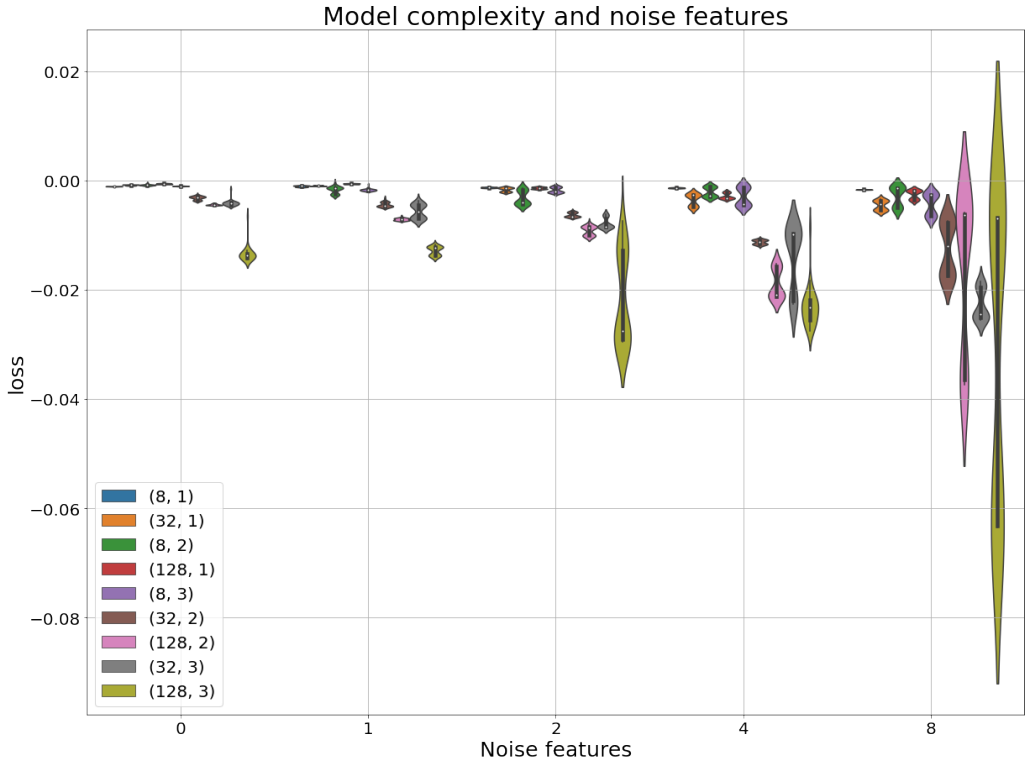

Figure 5.11: Comparison of PDP centred attacks' losses on different versions of `heart1`.

vation regards actual model complexity. It seems that the greater number of parameters neural network has, the more vulnerable model is. Thirdly, in almost all models in almost all dataset versions, there are two solutions found by the attack. One solution has a slightly lower loss function value.

Figures 5.12 and 5.13 show distribution of losses of robustness check on `friedman1` and `heart1` datasets respectively. The former plot is the hardest to interpret because it has the least distinctive features comparing to others. However, one may still find that the greater number of noise variables, the lower losses and the more complex model, the lower losses as well. On the other hand, Figure 5.13 is easy to describe. In both plots, we can observe two major solutions found by the model represented as a bimodal distribution. These results give the same conclusions as ones from Figures 5.10 and 5.11.

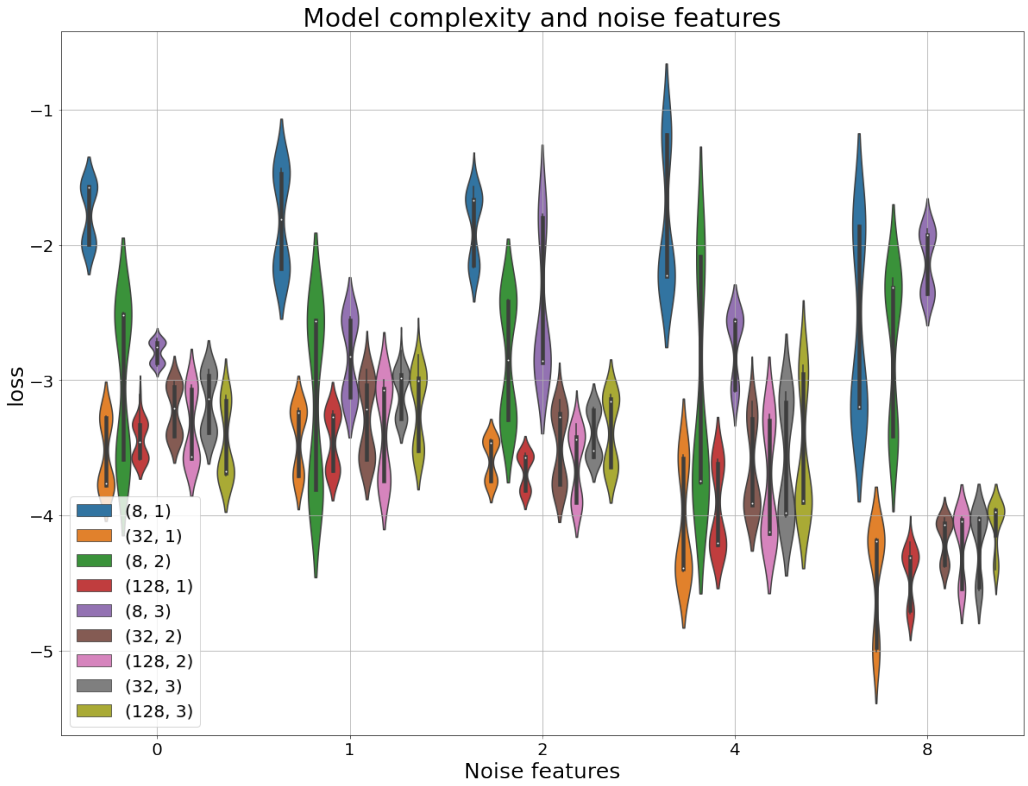

Figure 5.12: Comparison of PDP robustness checks' losses on different versions of `friedman1`.

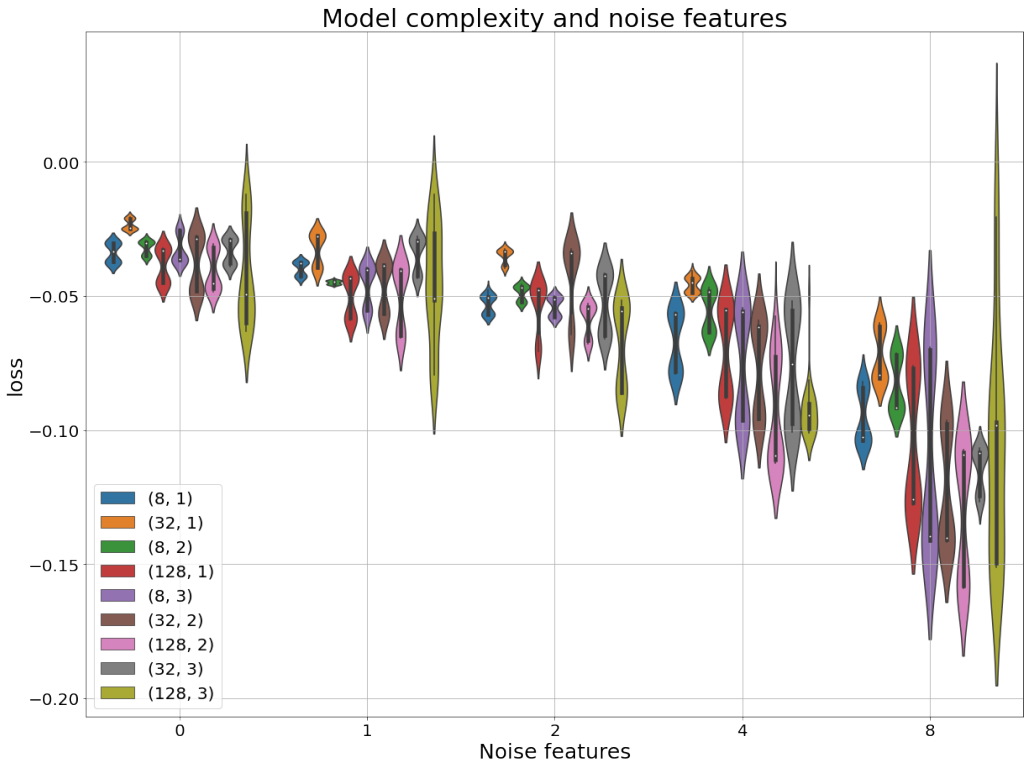

Figure 5.13: Comparison of PDP robustness checks' losses on different versions of `heart1`.

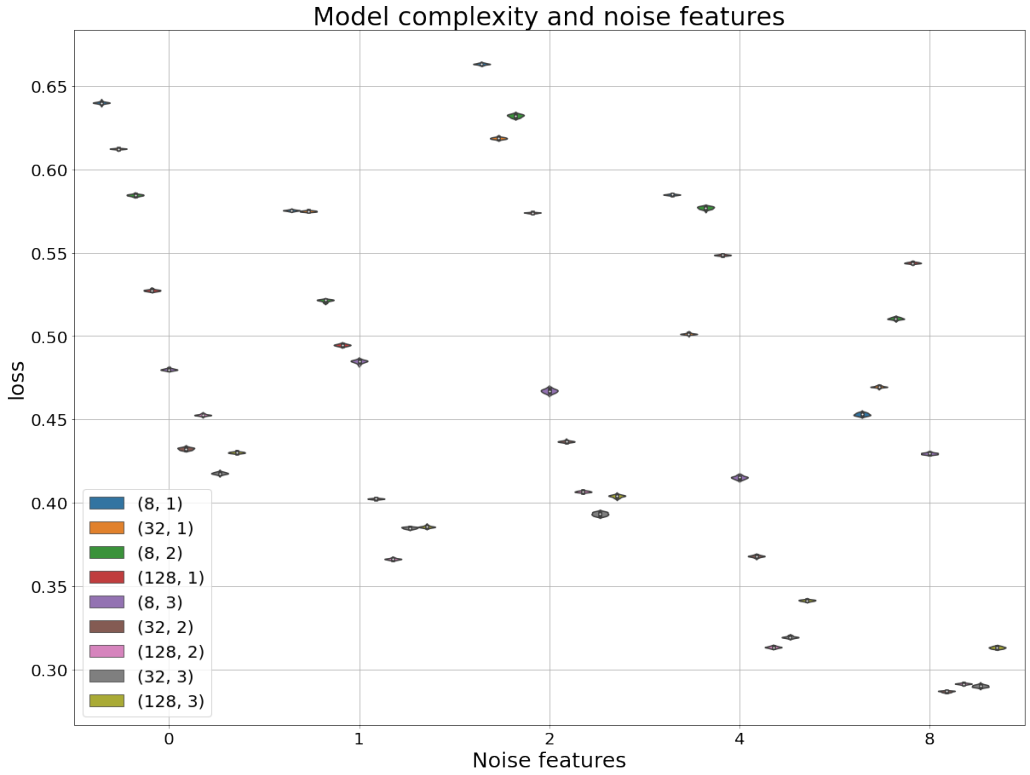

Figure 5.14: Comparison of PDP targeted attacks' losses on different versions of `friedman1`

A very distinctive feature of Figures 5.14 and 5.15 and targeted attacks, in general, is their lack of variability of solutions. In all of these examples/setups, for all architectures, for all different versions of the `friedman1` and `heart1` datasets, final loss in each group is basically the same. That ensures us that using targeted attack gives us always the best possible result in terms of loss function optimization. In previous examples depending on the random seed, we may obtain different results, and it is generally a good idea to run the algorithm multiple times to find the best solution. The targeted attack does not have this variance. It is hard to say if a greater number of added noise variables makes the model more vulnerable. However, similarly to previous results, we can again observe the situation, where more complex models are more vulnerable. Attacks on models with a greater number of parameters result with a lower value of loss function.

Final losses of attacks on ALE have very similar behaviour to ones of attacks on PDP in general. Figures 5.16 and 5.17 show robustness check final losses on `friedman1` and `heart1` datasets respectively. The greater number of noise variables, the lower final loss. Similarly the greater complexity of the model, the better solution (in terms of loss function). We can also observe two major solutions found by the model; they, are represented by a bimodal distribution. It may be a good idea to run an attack many times to obtain the best/most interesting result.

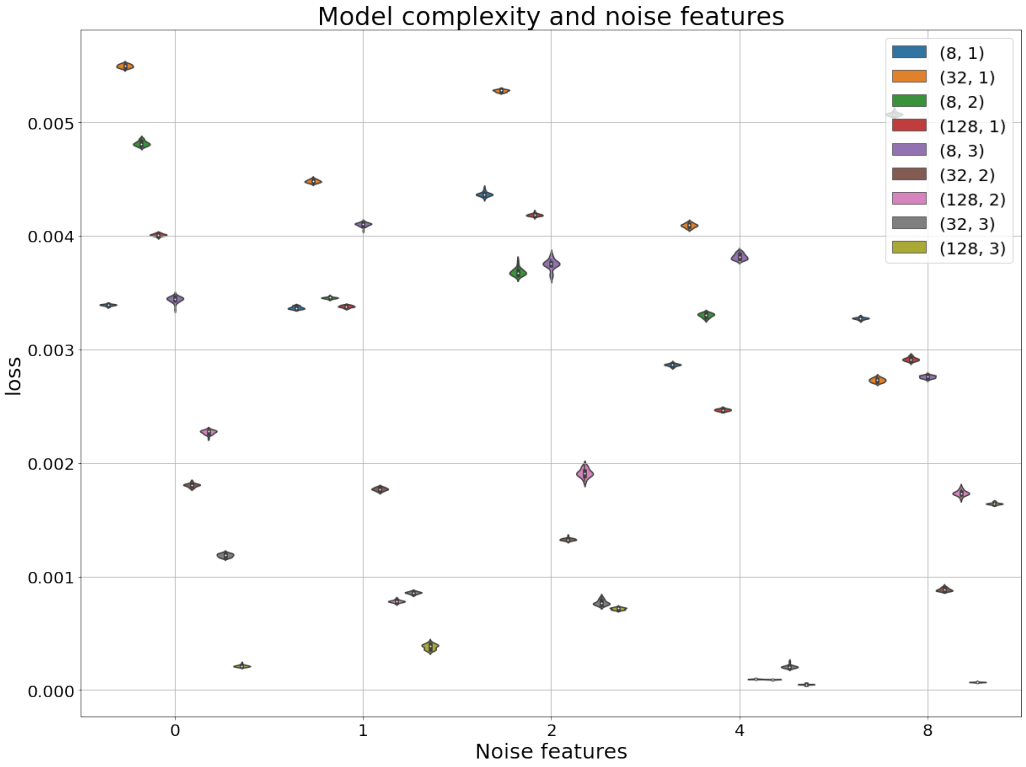

Figure 5.15: Comparison of PDP targeted attacks' losses on different versions of `heart1`.

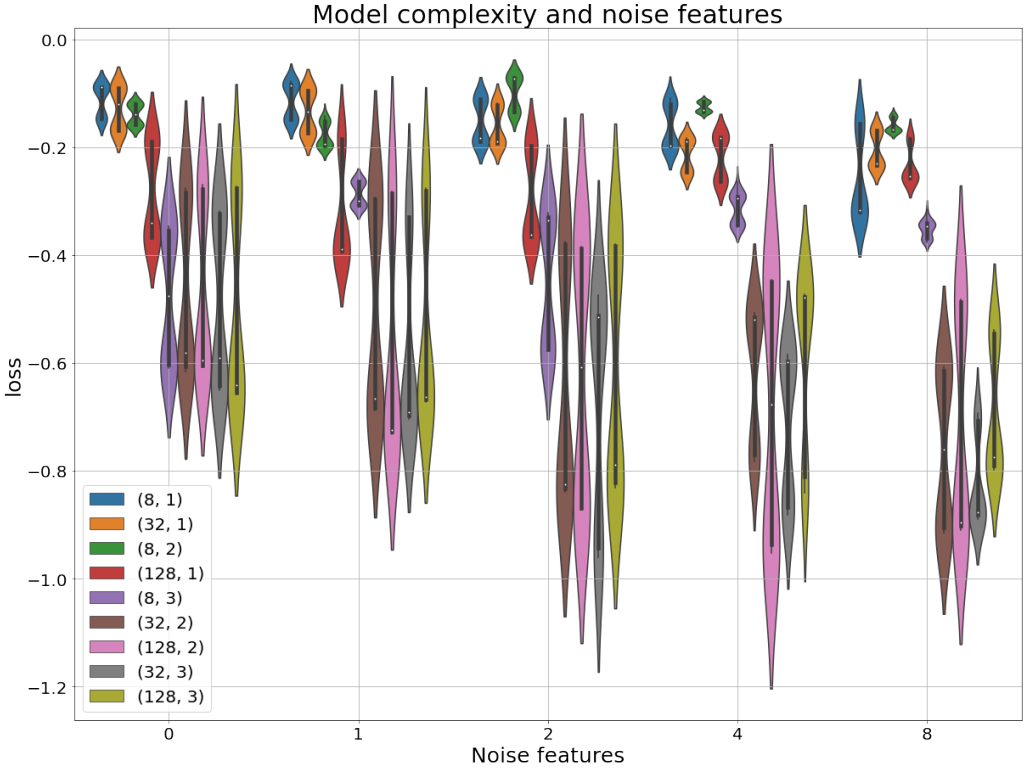

Figure 5.16: Comparison of ALE robustness checks' losses on different versions of `friedman1`.

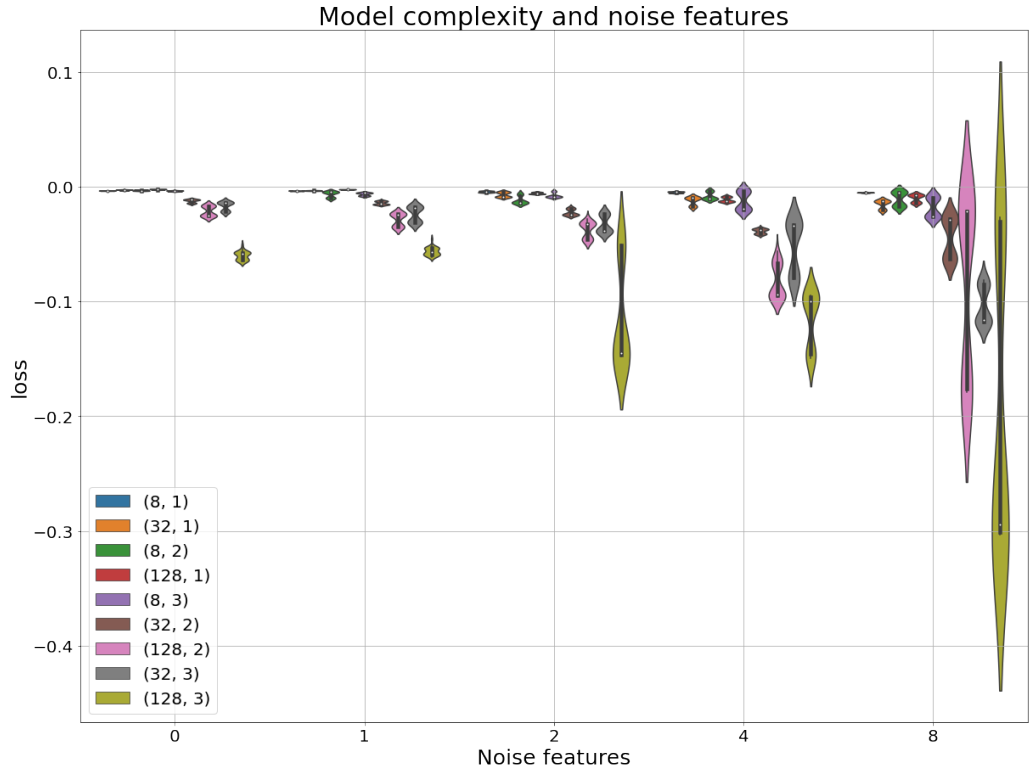

Figure 5.17: Comparison of ALE robustness checks' losses on different versions of `heart1`.

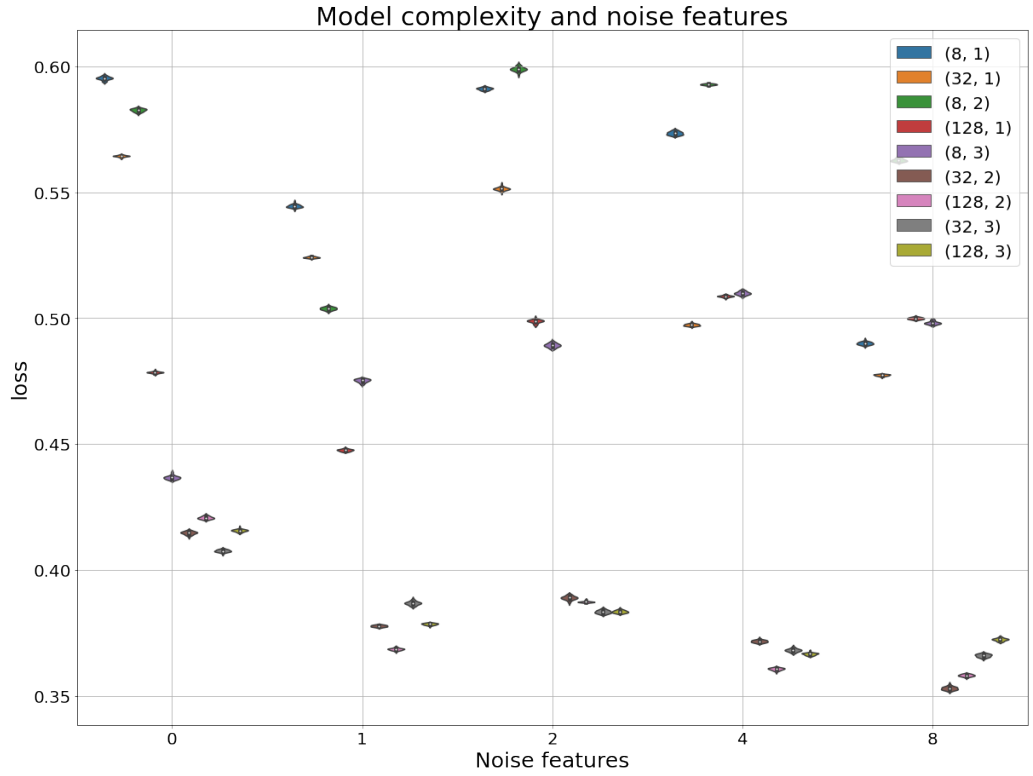

Figure 5.18: Comparison of ALE targeted attacks' losses on different versions of `friedman1`.

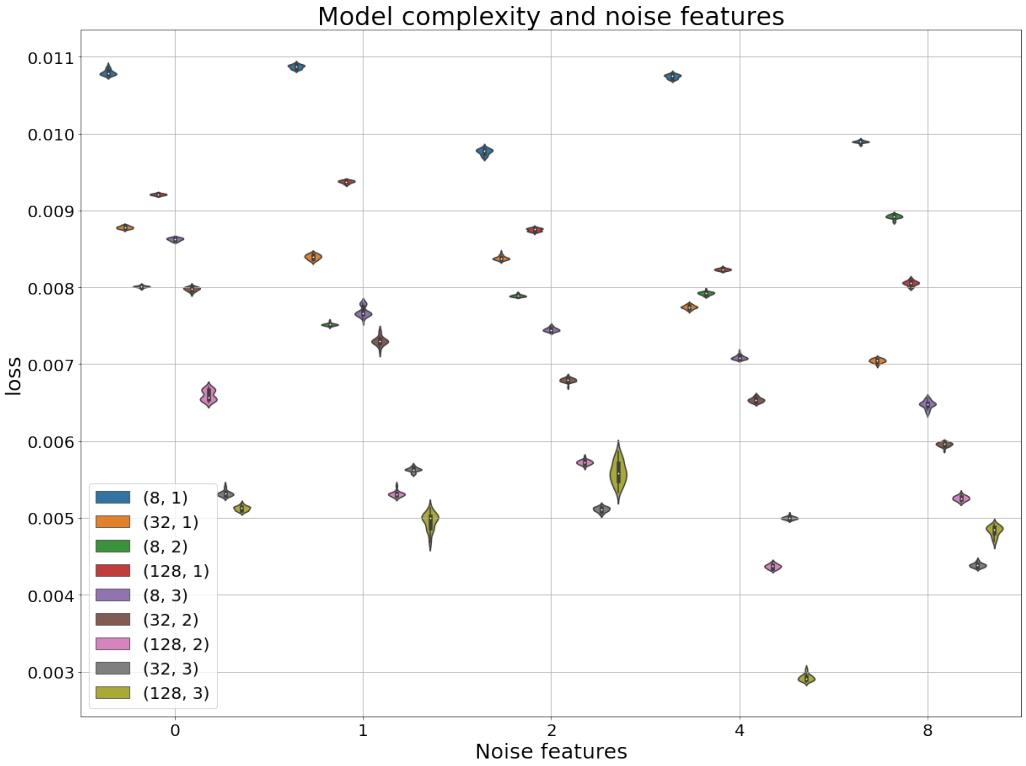

Figure 5.19: Comparison of ALE targeted attacks' losses on different versions of `heart1`.

Figures 5.18 and 5.19 presents final losses' distributions of the targeted attack on the ALE method. Similarly to targeted attacks on PDP, there is very little variance in final losses in each group. We may expect that a single run of an attack will give us the best solution closest to the target function. Just as in case of targeted PDP attacks is hard to say if a larger number of noise features makes the model more vulnerable. However, as previously, a greater number of parameters makes the model more vulnerable.

5.4. Analysis of data shift

In this section, we apply the algorithms following the framework presented in Figure 3.1 and analyze the data shift occurring in the underlying dataset after the attack. For that, we consider a simple predictive task on the `heart1` dataset. We aim to classify healthy individuals denoted as `1` in the target, as this is a one-versus-all prediction where several disease outcoms are possible (denoted for simplicity as `0`).

5.4.1. Targeted attack on PDP using genetic-based algorithm

Goal: Consider a data scientist training a machine learning model for the given classification task. The aim is to achieve a targeted visual result of model explanation to *provide* rationale for a given thesis. In this case, the desired outcome is an increasing relationship between the `age` variable and the expected value of models' prediction.

Model: Fitted model is a basic implementation of the SVM algorithm predicting a probability of being healthy (`AUC>0.9`). The PDP explanation shows a straight line on the decision boundary for the `age` variable (see the red line in the left panel of Figure 5.20). This may be interpreted as no relationship between the variable and models' predictions in a global-level sense.

Adversary: To achieve the goal of explanation manipulation, the data scientist uses a genetic-based targeted attack on PDP of the `age` variable. Categorical variables are set to constant and the change in data is constrained to the original range of variable values.

Results: Figure 5.20 presents the result of an attack changing only 4 out of 13 explanatory variables from the validation dataset. We can observe a drastic shift of distribution in the `thalach` variable, which partially divides the changed dataset from the original one. At the same time, Figure 5.21 presents the result in explanation, wherein the right panel we see the target, and in the left, the axis is brought closer. Data scientist achieves a change in monotonicity of the visual explanation due to data shift of variable `thalach`. Looking only at the right panel of Figure 5.21, one may provide a ground for a new thesis that the expected prediction of the model increases with `age` values.

Discussion: We call the potential of such attacks a security breach in explainability use. Explainable AI methods were supposed to bring transparency to black-box machine learning; however, when the model depends on numerous variables with interactions, its explanations may behave like black-box as well.

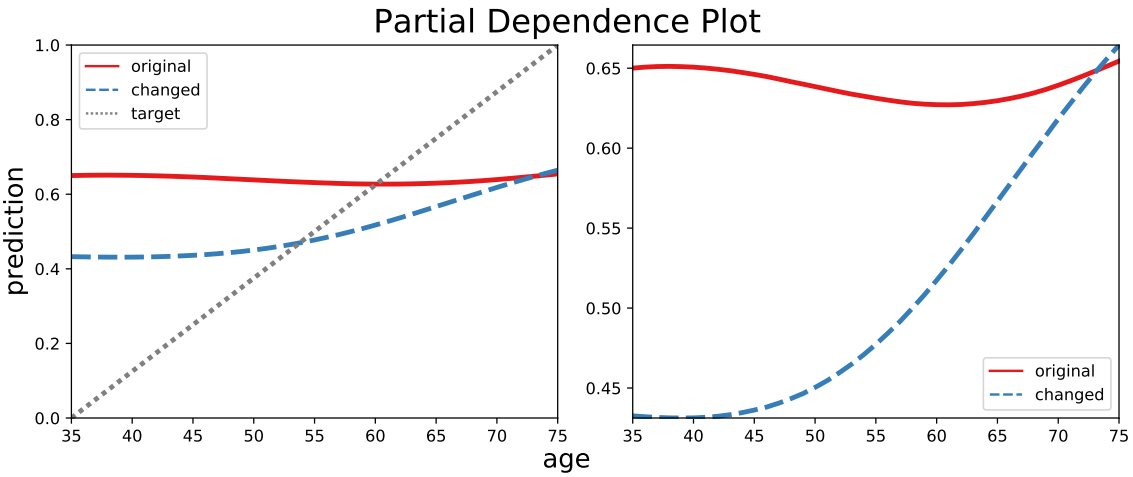

Figure 5.20: Result of the genetic-based attack on the PDP explanation of the `age` variable (data: `heart1`; model: SVM).

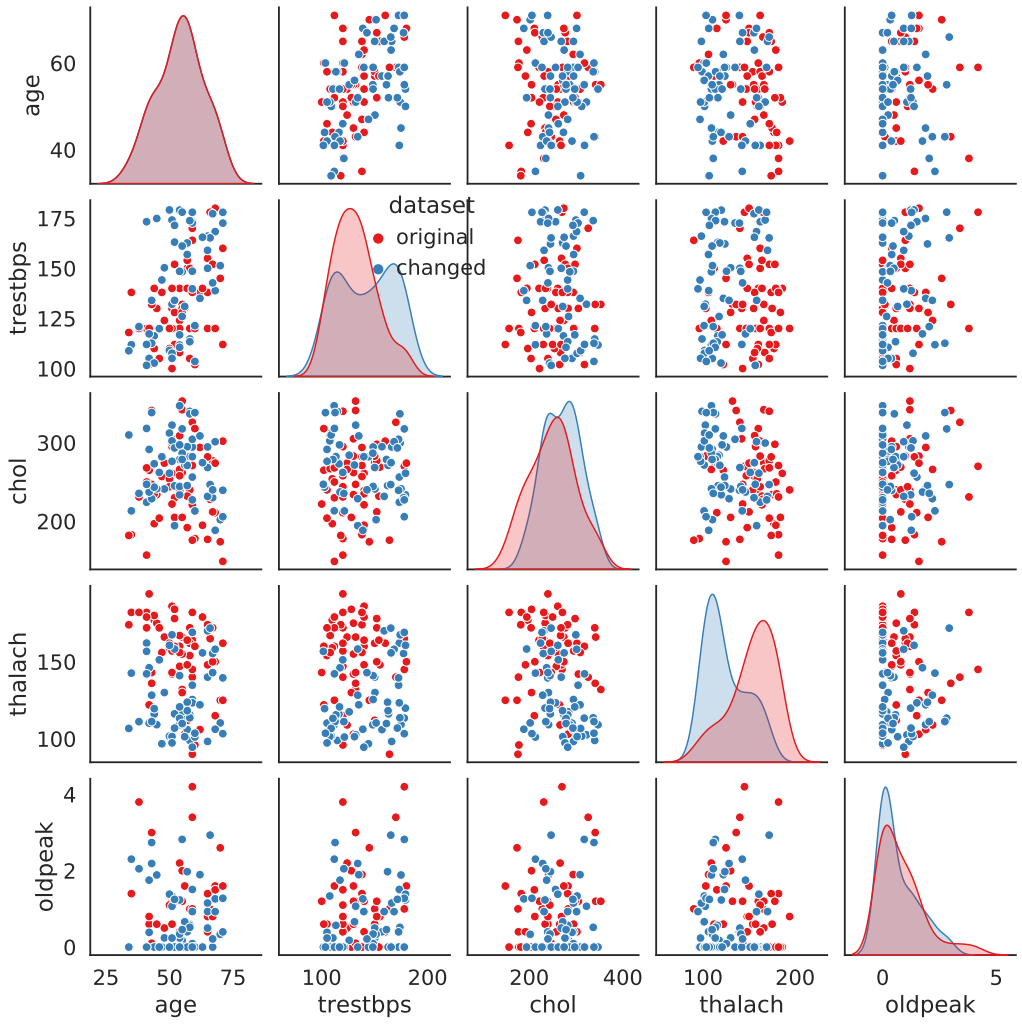

Figure 5.21: Result of the genetic-based attack changing 4 out of 13 explanatory variables (`age` and remaining variables are constant). Data shift can be tuned using the α parameter to regularize the loss function – for this example we set α to `0`. In this case we added constraints on the variable values not to exceed minimum and maximum of the original range.

5.4.2. Robustness check for ALE using gradient-based algorithm

Goal: Consider a data scientist training a machine learning model for the given classification task. The aim is to showcase a visual result of model explanation to *support* the rationale for a given thesis. In this case, the desired outcome is a robust relationship between the `age` variable and the expected value of models' prediction.

Model: Fitted model is a two-layer neural network predicting a probability of being healthy (`AUC>0.9`). The ALE explanation shows a nearly steady line for the `age` variable (see the red line in Figure 5.22). This may be interpreted as little relationship between the variable and models' predictions in a global-level sense.

Check: To achieve the goal of explanation evaluation, the data scientist uses a gradient-based robustness check for ALE on the `age` variable. Categorical variables are set to constant with no additional constraints.

Results: Figure 5.23 presents the result of an attack changing only 4 out of 13 explanatory variables from the validation dataset. We can observe mild shifts in distributions of `oldpeak` and `chol` variables. At the same time, Figure 5.22 presents the result in explanation, wherein there is no target due to the strategy used. Data scientist achieves a change in monotonicity of the visual explanation due to mild data shift of a few variables. Looking at Figure 5.22, the result may be a ground for undermining the stated thesis. Moreover, one could provide a ground for a new thesis that the expected prediction of the model decreases with `age` values; thus, the explanation is not robust.

Discussion: We provide the implemented attacks which can be used for the evaluation of PDP and ALE robustness. This is especially crucial when seeking guidance and providing evidence for some rationale through the explanatory analysis of complex models.

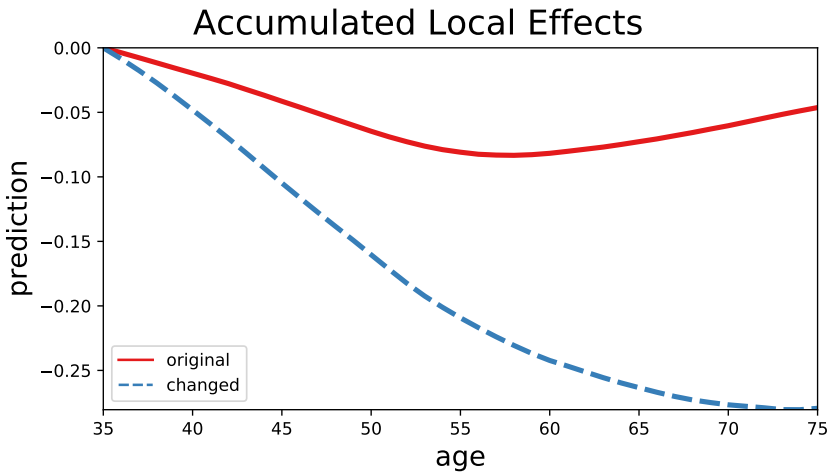

Figure 5.22: Result of the gradient-based attack on the ALE explanation of the `age` variable (data: `heart1`; model: two-layer neural network).

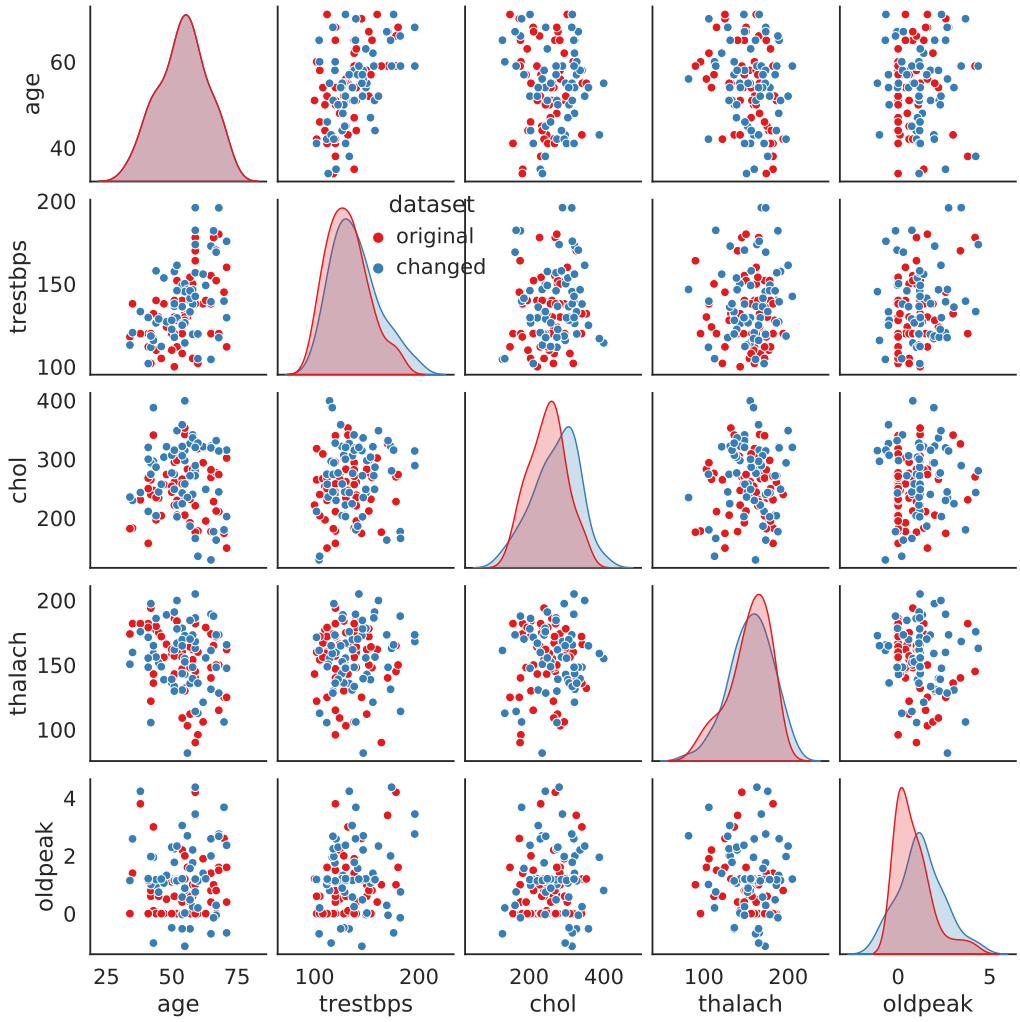

Figure 5.23: Result of the gradient-based attack changing 4 out of 13 explanatory variables (`age` and remaining variables are constant). Data shift can be tuned using the α parameter to regularize the loss function – for this example we set α to `0`. Some values may shift out of distribution e.g. to negative values, which is a possible improvement for the future works.

6. Conclusions

This work investigates the robustness of global-level, post-hoc model explainability from the adversarial setting standpoint, which refers to the responsibility and security of artificial intelligence use. We defined an optimization task, which aims at changing the visual model explanation through the poisoning in the underlying dataset used for the estimation. Possible manipulation of PDP and ALE leads to the conclusion that explanations used to explain black-box machine learning may be considered black-box themself. These Explainable AI methods are undeniably useful through implementations in various popular software. However, just as machine learning models cannot be developed without extensive testing and understanding of their behaviour, their explanations cannot be used without critical thinking. We recommend ensuring the reliability of the explanation results through the introduced attacks, which can also be used to study models behaviour under the data shift.

6.1. Summary

From the experiments, we have the following main results:

1. Both PDP and ALE curves can be strongly bent and shifted.
2. Explanations of tree-ensemble models showcase robustness to the attacks.
3. Support vector machine should not be explained with PDP and ALE methods, as the data poisoning easily manipulates these.
4. Targeted gradient-based attacks do not have variance in solutions, meaning they converge to the best possible one in terms of the loss function.
5. Robustness checks give varied results depending on the setting, e.g. may propose two opposite solutions, which is why it is advised to perform checks multiple times.
6. Data shift occurring due to the attack can be investigated to provide evidence of manipulation, or study the possible interactions in models.

6.2. Future work

We foresee several possible directions for future works. The proposed attacks consist of multiple parameters, e.g. regularization terms, that can be tuned to achieve more substantial fooling. Additionally, some may be interested in data poisoning of categorical variables, not only continuous space which we consider. We are also interested in a specific investigation of multiple solutions that these attacks can give to each task, as various optima may relate or differ extensively. Overall, the landscape of global-level, post-hoc model explanations is a broad domain and the potential of a security breach in other methods should be further examined.

Division of Work

This thesis is a joint work conducted by Hubert Baniecki and Wojciech Kretowicz. While the conceptualization and methodology involved both authors, we distinguish the division of work in Table 6.1.

Table 6.1: Division of work.

Task	**H.B.**	**W.K.**
Joint work	Conceptualization, Begining of Chapters 3 & 4, Python package	
Thesis text (mainly)	Chapters 1, 3.2, 5.4, Bibliography	Chapters 2, 3.3, 4.4
Implementation	Genetic-based attack	Gradient-based attack
Experiments	Genetic-based attack	Gradient-based attack

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265—283.

Adadi, A. and Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*, volume 31, pages 9505–9515. Curran Associates, Inc.

Adebayo, J., Muelly, M., Liccardi, I., and Kim, B. (2020). Debugging Tests for Model Explanations. In *Advances in Neural Information Processing Systems*, volume 33, page TBA. Curran Associates, Inc.

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K.-R., Dähne, S., and Kindermans, P.-J. (2019). iNNvestigate Neural Networks! *Journal of Machine Learning Research*, 20(93):1–8.

Alvarez Melis, D. and Jaakkola, T. (2018). Towards Robust Interpretability with Self-Explaining Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7775–7784. Curran Associates, Inc.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. (2018). Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In *6th International Conference on Learning Representations*.

Angelov, P. and Soares, E. (2020). Towards explainable deep neural networks (xDNN). *Neural Networks*, 130:185 – 194.

Apley, D. W. and 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.

Arik, S. O. and Pfister, T. (2020). TabNet: Attentive Interpretable Tabular Learning. *arXiv:1908.07442*.

Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., MojsiloviÄ‡, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J. T., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K. R., Wei, D., and Zhang, Y. (2020). AI Explainability 360: An Extensible Toolkit for Understanding Data and Machine Learning Models. *Journal of Machine Learning Research*, 21(130):1–6.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):1–46.

Baniecki, H. and Biecek, P. (2019). modelStudio: Interactive Studio with Explanations for ML Predictive Models. *Journal of Open Source Software*, 4(43):1798.

Baniecki, H., Kretowicz, W., Piatyszek, P., Wisniewski, J., and Biecek, P. (2020). dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python. *arXiv:2012.14406*.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82 – 115.

Bhatt, U., Weller, A., and Moura, J. M. F. (2020). Evaluating and Aggregating Feature-based Model Explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3016–3022.

Biecek, P. (2018). DALEX: Explainers for Complex Predictive Models in R. *Journal of Machine Learning Research*, 19(84):1–5.

Biecek, P. and Burzykowski, T. (2021). *Explanatory Model Analysis*. Chapman and Hall/CRC, New York.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934*.

Boopathy, A., Liu, S., Zhang, G., Liu, C., Chen, P.-Y., Chang, S., and Daniel, L. (2020). Proper Network Interpretability Helps Adversarial Robustness in Classification. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1014–1023. PMLR.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, page TBA. Curran Associates, Inc.

Chen, J., Vaughan, J., Nair, V., and Sudjianto, A. (2020). Adaptive Explainable Neural Networks (Axnns). *SSRN Electronic Journal*.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C., Mincu, D., Mitani, A., Montanari, A., Nado, Z., Natarajan, V., Nielson, C., Osborne, T. F., Raman, R., Ramasamy, K., Sayres, R., Schrouff, J., Seneviratne, M., Sequeira, S., Suresh, H., Veitch, V., Vladymyrov, M., Wang, X., Webster, K., Yadlowsky, S., Yun, T., Zhai, X., and Sculley, D. (2020). Underspecification Presents Challenges for Credibility in Modern Machine Learning. *arXiv:2011.03395*.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, volume 32, pages 13589–13600. Curran Associates, Inc.

Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43 – 53.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–67.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5):1189–1232.

Fukuchi, K., Hara, S., and Maehara, T. (2020). Faking Fairness via Stealthily Biased Sampling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):412–419.

Ghorbani, A., Abid, A., and Zou, J. (2019). Interpretation of Neural Networks Is Fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688.

Gill, N., Hall, P., Montgomery, K., and Schmidt, N. (2020). A Responsible Machine Learning Workflow with Focus on Interpretable Models, Post-hoc Explanation, and Discrimination Testing. *Information*, 11(3):137.

Greenwell, B. M. (2017). pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1):421–436.

Hancox-Li, L. (2020). Robustness in Machine Learning Explanations: Does It Matter? In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 640—647. Association for Computing Machinery.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Heo, J., Joo, S., and Moon, T. (2019). Fooling Neural Network Interpretations via Adversarial Model Manipulation. In *Advances in Neural Information Processing Systems*, volume 32, pages 2925–2936. Curran Associates, Inc.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. (2019). A Benchmark for Interpretability Methods in Deep Neural Networks. In *Advances in Neural Information Processing Systems 32*, pages 9737–9748. Curran Associates, Inc.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154. Curran Associates, Inc.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and sayres, R. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (Un)reliability of Saliency Methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer International Publishing.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

Konstantinov, A. V. and Utkin, L. V. (2020). Interpretable Machine Learning with an Ensemble of Gradient Boosting Machines. *arXiv:2010.07388*.

Lakkaraju, H. and Bastani, O. (2020). "How Do I Fool You?": Manipulating User Trust via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79—85. Association for Computing Machinery.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lipton, Z. C. (2018). The Mythos of Model Interpretability. *Queue*, 16(3):31—57.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc.

Merrer, E. L. and Trédan, G. (2020). Remote explainability faces the bouncer problem. *Nature Machine Intelligence*, 2:529—539.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Molnar, C., Casalicchio, G., and Bischl, B. (2018). iml: An R package for Interpretable Machine Learning. *Journal of Open Source Software*, 3(26):786.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv:1909.09223*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ribeiro, M. T., Singh, S., and 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*, pages 1135—1144.

Rieger, L. and Hansen, L. K. (2019). Aggregating explanation methods for stable and robust explainability. *arXiv:1903.00519*.

Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1:206–215.

SauceCat (2018). *PDPbox: python partial dependence plot toolbox*. v0.2.0.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *In 2nd International Conference on Learning Representations*.

Slack, D., Hilgard, S., Jia, E., Singh, S., and 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*, pages 180—186. Association for Computing Machinery.

Sudjianto, A., Knauth, W., Singh, R., Yang, Z., and Zhang, A. (2020). Unwrapping The Black Box of Deep ReLU Networks: Interpretability, Diagnostics, and Simplification. *arXiv:2011.04041*.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Wang, Z., Wang, H., Ramkumar, S., Mardziel, P., Fredrikson, M., and Datta, A. (2020). Smoothed Geometry for Robust Attribution. In *Advances in Neural Information Processing Systems*, volume 33, page TBA. Curran Associates, Inc.

Warnecke, A., Arp, D., Wressnegger, C., and Rieck, K. (2020). Evaluating Explanation Methods for Deep Learning in Security. In *2020 IEEE European Symposium on Security and Privacy*.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61.

Wright, A. H. (1991). Genetic Algorithms for Real Parameter Optimization. In *Foundations of Genetic Algorithms*, volume 1, pages 205 – 218. Elsevier.

Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., and Wang, T. (2020). Interpretable Deep Learning under Fire. In *29th USENIX Security Symposium*, pages 1659–1676. USENIX Association.

List of symbols and abbreviations

AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
PDP	Partial Dependence Plot
ALE	Accumulated Local Effects
Adam	Adaptive moment estimation

List of figures

List of tables