TU WIEN Informatics

Erklärung der Unterschiede in den Entscheidungsgrenzen trainierter Klassifikationsmodelle

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Karl Maier, BSc
Matrikelnummer 01426356

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Wien, 1. September 2022

Karl Maier	Andreas Rauber

TU WIEN Informatics

Explaining the Differences of Decision Boundaries in Trained Classifiers

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Karl Maier, BSc
Registration Number 01426356

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Vienna, 1st September, 2022

Karl Maier	Andreas Rauber

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Karl Maier, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. September 2022

Karl Maier

Kurzfassung

Seit dem Aufkommen der Black Box Society [Pas15] werden die Entscheidungen von Modellen im Bereich des Maschinellen Lernens zunehmend unverständlicher wegen ihrer hohen Komplexität. Stattdessen muss man sich beim Vergleichen von Modellen auf Gütekriterien verlassen, die jedoch eine zu ungenaue Abbildung der Realität sind [DVK17]. Weiters werden die externen Anforderungen immer höher, Entscheidungen transparent darzulegen. Mit XAI (Deutsch: erklärbare künstliche Intelligenz) Methoden kann man Einsicht in die Vorgänge eines Modells erlangen. Darunter ist SHAP [LL17] als Methode besonders interessant, da sie aufgrund ihrer speziellen Eigenschaften erlaubt, eine Erklärung selbst als Modell zu behandeln. Aber es gibt noch wenige Ansätze, um Modelle zu vergleichen. Mit DiRo2C [Sta21] wurde direktes Erklärbar-Machen mittels eines Vereinigungsmodells für Klassifikationsmodelle vorgeschlagen. In dieser Masterarbeit stelle ich die Methode Mocca-SHAP vor, die SHAP verwendet um sowohl traditionelle Erklärungen als auch Erklärungen für das Vereinigungsmodell zu erzeugen. Mit Supervised Clustering [LEL19] werden die Erklärungen in eine modulare, hierarchische Struktur eingeteilt. In SHAP Dependence Plots [LEL19] lassen sich viele Erklärungen aggregiert darstellen und der Effekt eines Attributs auf eine Zielvariable interpretieren. Group Counterfactual Explanations unterstützen dabei, kausale Zusammenhänge zu verstehen. So tritt man mit den Erklärungen selbst in eine Konversation und interpretiert sie interaktiv. Ich evaluiere im Kontext der Endanwendung mittels kontrollierter Experimente, wo die zu erklärenden Unterschiede bekannt sind. Dabei kommen quantitative und qualitative Qualitätsmetriken zum Einsatz. Zu den Quantitativen zählen *Fidelity*, *Complexity* und *Generation Time*. Qualitativ evaluiere ich bezugnehmend auf Millers wünschenswerte Eigenschaften von Erklärungen [Mil19] *Contrastiveness*, *Selectiveness*, *Causality* und *Interactivity*, die zusammen *Contextuality* ausmachen.

Abstract

Since the rise of the black box society [Pas15] in the last decade, the decisions of machine learning models are increasingly less understandable because of their high complexity. Instead, data scientists have to rely on performance metrics when comparing models during development, selection or monitoring. But a single metric is a too simplistic description of most real-world tasks [DVK17]. Also, law makers increasingly require transparency of automated decisions. With interpretability methods in the field of XAI one can gain insights into the inner workings of a single model. Among them, SHAP [LL17] is especially interesting because of its special properties, that allow an explanation to be treated as a model. But there is a lack of systematic ways of comparing multiple models. With DiRo2C [Sta21], differences are sought to be explained directly with the difference classifier, an intermediate model, that merges the outputs of two classifiers. In this master thesis I propose Mocca-SHAP, which uses SHAP to create traditional explanations and explanations for the difference classifier. Supervised clustering [LEL19] creates modular, hierarchical groupings of explanations. SHAP Dependence Plots [LEL19] aggregate many explanations to allow interpretation of how a feature affects an outcome. Group Counterfactual Explanations aid in understanding causal relations. With this, one can interact in a conversational way with the explanations. I evaluate in the context of the end task in controlled experiments with a priori known differences that are to be found on a variety of quantitative and qualitative quality metrics. Quantitative metrics include *Fidelity*, *Complexity* and *Generation Time*. Qualitatively, I evaluate with regard to Miller's desirable properties for explanations [Mil19] *Contrastiveness*, *Selectiveness*, *Causality* and *Interactivity*, which together make up *Contextuality*.

Contents

CHAPTER 1

Introduction

Because of the rise of the black box society in the last decade [Pas15], data scientists are increasingly unaware of the reasons of the predictions of their machine learning models. A black box model is a model, that makes predictions without explaining why. Other models can be interpretable by looking at their inner weights, like linear models, which are termed inherently interpretable models. In the field of Explainable Artificial Intelligence (XAI), one seeks to explain the behaviour post-hoc of a black box with interpretability methods. In this master thesis, I focus on the goal of understanding the differences in classifiers, which are a type of predictive machine learning model.

This section explains, why there is a need for comparing machine learning models not only by using traditional performance metrics but also interpretability methods. Further, existing work is described and how this research aims to help. Then I describe the structure of this master thesis.

1.1 Problem Statement

1.1.1 Motivation

1.1.2 Iterative Model Development

The process of developing supervised machine learning models is iterative, for example when employing a standard process model like the CRoss Industry Standard Process for Data Mining (CRISP-DM) [WH00]. When incrementally refining predictive models, there is a need for comparing them. This is usually done by measuring performance, like accuracy or Root Mean Square Error (RMSE). At the end of development, a selection is done which also requires comparison [SSSEA19]. However, Doshi-Velez and Kim [DVK17] argued, that a single metric is a too simplistic description of most real-world tasks. Therefore, decisions are based on an incomplete explanation of their performance. They state that the criterion of choice in this case is interpretability.

Because of the *Rashomon effect* there might be two models with equal performance, but different inner workings [B+01]. Interpretability can be used to uncover such differences [BB21, p. 266].

Even after deployment, there is a need to constantly compare models when new data is available, because one model could perform better than another if concept drift occurred [Nat07].

1.1.3 Right to Explanation

Regulators are also addressing the problem of transparency and hidden biases with the General Data Protection Regulation (GDPR) [Eur16, PGG+19]. More recently, the European Commission proposed the *"Artificial Intelligence Act"* [Eur21], which takes transparency one step further. The requirements will be stricter for applications interacting with humans, applications that detect emotions, categorize based on biometric data or transform content. Doshi-Velez and Kim further note, that *"interpretability can assist in qualitatively ascertaining whether other desiderata — such as fairness, privacy, reliability, robustness, causality, usability and trust — are met"* [DVK17].

1.1.4 Problem Statement

Data scientists are facing the aforementioned problems of limited expressiveness of traditional performance metrics for model comparison and increasingly pervasive laws. They further require a comparison method to be model-agnostic, because they may want to choose from models of different types during model selection. Therefore, there is a need for global understanding of the differences between models which can be satisfied with model-agnostic interpretability methods.

1.2 Existing Work

This section gives an overview about existing work on model comparison using interpretability and focusing on global model understanding rather than just instance-level understanding.

Most interpretability methods are designed to explain a single model. With these, one has to generate explanations for each model to be compared separately and then try to interpret what is different.

Explanations in their simplest form are feature importances and are also used to compare models [ZWM+18, BB21].

More detailed methods that visually explain a model's input-output relation are also used to compare models [KPN16, BB21]. Biecek and Burzykowski give an example of how to compare using Partial Dependence (PD) plots [Fri01] by plotting the dependence curve for each model into the same graph [BB21, p. 213]. The authors further show an

instance-level example in which they use side-by-side force plots of SHAP Values [LL17] to compare multiple explanations of different models [BB21, p. 164].

A more direct approach is taken by Staufer and Rauber with Difference Recognition of 2 Classifiers (DiRo2C). They define an intermediate model that combines the target models' predictions and apply an improved version of LOcal Rule-based Explanations (LORE) [GMR+18a] on it to explain at the instance-level, but also propose a global-level extension method. [Sta21]

Except for DiRo2C, I found no other works that have a direct focus on model comparison with interpretability and treat the models as black-boxes.

1.3 Solution

I propose the new method Model comparison with clustered difference classifier SHAP values (Mocca-SHAP). It is focused on classifiers, just as DiRo2C [Sta21] does and builds on the same idea of explaining the outputs of the difference classifier. In contrast, it uses SHapley Additive exPlanations (SHAP) [LL17] as the underlying method along with its global-level extensions and combines traditional explanations for individual classifiers and explanations for the difference classifier in one method. With that, it can be used to compare any kind of classifier because it treats them as black-box models.

1.3.1 Research Questions

Given a data set to be explained, where some instances are classified differently and two classifiers, I address the research question:

To what extent can SHAP Values, generated for a reformulated difference classification task, improve the process of understanding, why certain instances in a data set are classified differently by the given classifiers?

In order to achieve that, I break it down into smaller research questions:

1. What are appropriate quality metrics on which to compare different model comparison methods?
2. Given the difference classification problem formulation proposed with DiRo2C [Sta21], which of the possible variations (binary classification, full multiclass classification or partial multiclass classification) is suited best for the chosen interpretability method SHAP?
3. Which of the extensions methods of SHAP are suited best to facilitate global-level model understanding?
4. Which are appropriate means to create more targeted local explanations for groups of instances?

5. How does the proposed method perform in terms of the quality metrics specified compared to a baseline approach and DiRo2C in controlled experiments with artificial and benchmark tasks?

1.4 Research Contributions

With Mocca-SHAP I intend to address the sparse research in the area of model comparison using interpretability. In the long run, it should enable data scientists to achieve their goals better by providing an alternative which explicitly focuses on comparison. It should support them in all stages of a data science process: (1) during iterative development to identify strengths and weaknesses early for potential optimizations, (2) during model selection, because the *Rashomon effect* [B+01] may have unknown consequences, (3) after model deployment, to identify how models respond to data drift.

1.5 Structure of the Thesis

This work is organized as follows. In Chapter 2, I explain the basic concepts such as predictive machine learning models, interpretability and explanations, give an overview about interpretability methods and model comparison methods with interpretability. In Chapter 3, I describe the methodology Design Science and how I apply it in this work. I also show how literature is reviewed and how metrics are selected for evaluating in controlled experiments. In Chapter 4, I describe three tasks based on artificial data sets and two tasks based on benchmark data sets, and what is to be expected from the explanations. In Chapter 5, I describe how the artifact is iteratively developed and assessed, how I automate metric calculation and share implementation details. In Chapter 6, I describe the experiments themselves and my findings. Finally, in Chapter 7, I summarize my findings and provide future research directions.

CHAPTER 2

Background

This chapter explains the basic parts needed to understand Mocca-SHAP. First, I explain which machine learning model types there are and which are in the focus of this thesis. I continue to define interpretability and what explanations are. Then I show different approaches at categorizing interpretability methods and explanations. Next, I give an overview about interpretability methods that are relevant for this work and go into detail about SHAP. Finally, I describe the traditional way of comparing models using interpretability and explain DiRo2C's more direct approach with the difference classifier.

2.1 Machine Learning Models

In this work I focus on predictive machine learning models. They are also called supervised machine learning models, and need a definition of a ground truth before the training phase in contrast to unsupervised machine learning models. There are basically two types of predictive models: [HTFF09]

- *Regression Model*: The predicted output $\hat{y}$ is a continuous number.
- *Classifier*: The predicted output $\hat{y}$ is a categorical label. The possible values are called classes. [HTFF09]

A data set X is used to train a supervised model, with a part reserved for evaluating its performance, which is called the test set. An explanation data set can be a separate data set or the training or test set and is used to generate explanations with an interpretability method. X is a matrix that contains the rows or instances $x^{(i)}$, with i being the number of the instance. Each instance has multiple values, one per feature j. The columns of X are denoted by x_j and a single value within the matrix is denoted by $x_j^{(i)}$. An interpretability

method requires as input a trained machine learning model with a prediction function $\hat{f}$ that estimates a target variable y with the prediction $\hat{y}$ (see Equation 2.1). [Mol20, p. 14]

$$\hat{f}(x^{(i)}) = \hat{y} \tag{2.1}$$

Probabilistic Classifiers

Probabilistic classifiers expose probability estimates for each class and predict the label with the highest class probability. Thus, they can be seen as regression models, that have an additional stage which determines the final label. Many classifier implementations (e.g. scikit-learn [PVG+11]) allow to predict probabilities based on internal measures that quantify how confident a classifier is about a prediction. As output, the classifier gives a vector with an estimate per class. The estimates are normalized, so that they sum up to 1. This information can be leveraged by interpretability methods that work with continuous targets. I distinguish two prediction spaces of classifiers: [BN06]

- *Probability space*: Its range is $[0, 1]$. If no internal probability estimate is available from a classifier, labels can be transformed to 0s and 1s with one-hot encoding to resemble probabilities. [BN06]

- *Log of odds space*: Probabilities can be transformed to odds ratios by dividing the probability of an event happening by the probability of an event not happening. Further taking the logarithm results in log of odds, also called log odds. The value range is a centered space around 0, which stretches on both sides to infinity. The transformation is also called the *logit* function, and defined as $logit(p) = log(\frac{p}{1-p})$ for probability p. It has the advantage that updating probabilities can be done by addition of log odds values instead of doing complex operations in probability space. This combines well with the additive feature attributions that are part of SHAP and is used heavily in the research of Lundberg et al. [LL17, LEL19, LEC+20]. But one needs to be cautious, because the lowest and highest probability values 0% and 100% map to $-\infty$ and ∞, and thus break further operations. [BN06]

The probability estimates may be distorted by characteristics of the classifier, which is not so much of a problem when comparing classifiers of the same type but could lead to biased explanations when comparing classifiers of different types. Platt scaling [Pla99] and Isotonic Regression [ZE02] can be used to calibrate probability estimates of a classifier. These methods are supported in recent machine learning frameworks[1]. [NMC05]

[1] scikit-learn Probability calibration: https://scikit-learn.org/stable/modules/calibration.html

Types of Classifiers

The simplest classifier is the binary classifier. It predicts a label that can present itself as one of two possible classes. Sometimes, they are referred to as the positive and negative label. In probability space, this can be described as a regression task to predict the probability of the positive label and a binarization stage with a cut-off value of 0.5. Or, it can be described as a multi-output regression task to estimate each class' probability and a classification stage, which predicts the class with the highest probability. The second form is necessary in case of multiclass classification, where the target includes more than two classes. [BN06, HTFF09]

2.2 Interpretability

According to Doshi-Velez and Kim [DVK17], interpretability means that a system can explain its reasoning, allowing it to be verified for soundness. Yet as they further explain, there is little consensus on what interpretability is. Miller uses Biran and Cotton's work [BC17] to define interpretability as *"the degree to which an observer can understand the cause of a decision."* [Mil19, p. 14]. Kim et al. define it differently as *"a method is interpretable if a user can correctly and efficiently predict the method's results"* [KKK16, p. 7]. I argue that a good explanation should serve both. Although the term interpretability is used more often, explainability is tightly tied to it, as Adadi and Berrada note, that *"interpretable systems are explainable if their operations can be understood by human"* [AB18, p. 5].

2.2.1 Explanations

The output of an interpretability method, ready for interpretation, is termed explanation in this work. Miller's major findings in his survey [Mil19] are, that there are four important properties for explanations:

- They should be *contrastive*. That means, that explanations state why something was predicted instead of something else. E.g. counterfactual explanations are contrastive, described in Section 2.3.
- They should be *selected*, which means, that humans do not want complete explanations, but rather one or two selected reasons.
- Showing *causal links* is more important than stating probabilities. An explanation may state the probability that it is true or use probabilities to explain. Both can be helpful, but probabilities alone have been found to be unsatisfying.
- Explanations should be *social*. This means that explanations are a transfer of knowledge from the explainer to the explainee, or an interaction. In this thesis, I refer to this property as *interactivity*. [Mil19]

These findings converge on one point: that explanations are *contextual*. The author explains in more detail, that *"While an event may have many causes, often the explainee cares only about a small subset (relevant to the context), the explainer selects a subset of this subset (based on several different criteria), and explainer and explainee may interact and argue about this explanation."* [Mil19, p. 6] This shall be the guiding principle while developing the artifact.

2.2.2 Taxonomy of Interpretability Methods

According to Carvalho, Pereira and Cardoso [CPC19], machine learning interpretability can be divided into several categories, which I describe in the next sections.

Intrinsic vs. Post Hoc

A model can be intrinsically interpretable by constraining its complexity. But in real world settings the best performing models are often ensemble models or very complex models such as deep neural networks which are not intrinsically interpretable [CPC19, p. 2]. Still, they have a use case in surrogate models, which are discussed in more detail in Chapter 2.3.

Post hoc interpretability refers to methods, that help in interpreting models after they are trained. They can be further distinguished into model-specific and model-agnostic methods. [CPC19]

Model-Specific vs. Model-Agnostic

If an interpretability method makes use of knowledge it extracts from a model's internals, such as learned weights, it is considered a model-specific method. If a method makes no assumptions on the underlying model and treats it as a black box, it is a model-agnostic method and can be applied to any model. In a process model like CRISP-DM, a model is incrementally developed. Also, it is repeatedly compared to its preceding version, which allows the use of both model-agnostic and model-specific methods. But if one wants to select one of several models of different types, only model-agnostic methods can be used. [CPC19]

In this thesis I choose to only include model-agnostic methods for maximum flexibility.

2.2.3 Scope of Interpretability

Carvalho, Pereira and Cardoso [CPC19] differentiate between interpretation on the global and the local level:

- *Global model interpretability:* In order for a model to be interpretable on a global level, the question *"how do parts of the model affect predictions"* [CPC19, p. 14] needs to be answered. I will describe an example for this in more detail in Section

2.3. This is a modular view. A holistic view is barely achievable in practice, because it would be necessary to understand the relation between data input and prediction space distribution in its entirety. [CPC19]

- *Local model interpretability:* Local model interpretability explains, why the model arrived at a specific prediction for a single instance, also called *instance-level interpretability*, or for a group of instances. Such an explanation has high local accuracy, and does not need to be accurate on a global level. [MCB20, CPC19]

Adadi and Berrada [AB18] further differentiate global model interpretability on a modular level, which explains only part of a model in contrast to global model interpretability.

Global explanations can also be created by accumulating local explanations. Pedreschi et al. [PGG+19, p. 8] describe such a procedure for the local explainers LORE [GMR+18a] and Anchors [RSG18]. SHAP dependence plots [LEL19, p. 8] work in a similar manner.

2.3 Overview about Interpretability Methods

Molnar distinguishes different types of possible results of interpretation methods [Mol20, p. 25], which I describe in the following sections. According to Adadi and Berrada [AB18, p. 17], amongst model-agnostic methods visualization is the most human-centered. So in order for statistics and other results to be the most comprehensible by humans, they should be visualized. [Mol20]

Feature Summary Statistics

In the simplest case, the result is a single number per feature to describe feature importances, sometimes called feature attributions [CPC19, p. 21]. It can be an absolute number, denoting the magnitude of importance, like the *Permutation Feature Importance* [B+01, FRD19]. It can also be a real number, denoting a specific increase or decrease of a target variable like it is the case with SHAP Values [LL17]. As a visualization technique, bar plots are commonly used. [Mol20]

A more complex result is a statistic which includes feature attributions for many feature values, like Partial Dependence (PD) [Fri01]. This method shows the marginal effect of a feature on the outcome. It can reveal, if the relationship between the features and the target is linear, monotonic or complex. The partial dependence function is estimated for a feature, and applied to every observed feature value. This results in pairs of feature values and average outcome values. It is best visualized as a scatter plot. Extension methods have been proposed that can explain a feature's importance and interactions [GBM18]. The main drawback is, that this method assumes independence of the features. Accumulated Local Effects (ALE) [AZ20] plots tackle this disadvantage and provide an unbiased result. They are also able to show interactions, but their interpretation requires more expertise. [Mol20]

Data Points

Example-based interpretability methods yield data points as a result, that explain the distribution of the model's output or local properties around an instance of interest. It only makes sense, if an instance's feature values themselves are meaningful as explanations. With a high number of features it may not be interpretable any more. [Mol20]

Most important for my work are *Counterfactuals* [WMR17, DMBB20]. These methods explain by stating instances, that are similar to an instance of interest but are predicted differently, telling you what needs to be changed for the output to change. The resulting instances are actually perturbed versions of the instance of interest. This satisfies the contrastiveness property of explanations, and by keeping the number of modifications low, they are also selective. [Mol20, WMR17, DMBB20]

Model Internals

Some models can be interpreted using knowledge about their internal behaviour. By definition, these methods are model-dependent. Even though I limit my research to model-agnostic methods, some knowledge about interpretability methods based on model internals is required. This is because methods for intrinsically interpretable models fall into this category, and they are used in conjunction with surrogate models, described in the next section. But I cover the basic types first: [Mol20]

- *Decision Tree*: A decision tree is generally considered to be interpretable, if it does not have too many nodes and levels. Starting from the root node, each node splits the data set by a certain feature and cut-off value. The leaf nodes are finally used to predict a label based on the label of the majority of training data instances ending up in it. This way, one can reproduce, how the model arrives at a decision. Decision trees allow to calculate feature importances from model internals via the Gini importance (also called Mean Decrease in Impurity) [B+01]. An advantage is, that a tree structure can capture interactions between features. Yet, trees fail to represent linear relationships, because they need to be approximated by a series of cut-off values in multiple nodes. They are also very sensitive to changes in the training data, meaning that slight changes might result in very different trees. [Mol20]
- *Decision Rules*: Decision rules state conditions and an outcome. They can be sets (unordered) or lists (ordered). A default rule defines what is predicted if no other rule applies. Just like decision trees, decision rules are bad at describing linear relationships. They can also be derived from a decision tree's internal structure. [Mol20, Sta21]
- *Linear Model*: A linear model can be interpreted by its learned weights. Each feature has an associated weight, that denotes its influence on the outcome. The biggest advantage of this method is linearity. Furthermore, the effects are additive,

meaning that they add up to the final outcome. It is also possible for a linear model to incorporate an interaction term, to explain interactions. Still, with strongly correlated features it is problematic to estimate the weights because it may not be possible to determine to which of the features the effect attributes. Another disadvantage is that the standard linear regression model assumes normality of the target, but this can be tackled with generalized linear models. The only relations, that it can model, are linear relations. But with transformations, it is possible to model other relations. [Mol20]

Surrogate Models

A black-box model can be explained by another model, which is an intrinsically interpretable model and has been trained to model the predictions of the black-box model. This is called a surrogate model. By seeing the simpler model as a proxy for the complex model, one can draw conclusions about its behaviour. When training the surrogate model, its fit to the predictions of the black-box model is optimized, instead of the true labels of the dataset, because the goal is to understand the behaviour of the black-box model, not the relation between the features and the target. Such an interpretability method is described by Hall et al. [HGKP20, p. 11]. Of course, it is not possible to understand the feature-target relation of any sufficiently complex black-box model by a surrogate model of limited complexity. But it can provide an overview for the data scientist at work. This disadvantage does not hold on the local level, when the goal is to understand the behaviour around an instance of interest. [HGKP20, Mol20]

Examples of methods based on surrogate models include:

- *Local Interpretable Model-agnostic Explanations (LIME)* [RSG16]: Ribeiro et al. proposed a method that focuses on local surrogate models. In order to train this local model, they proposed an algorithm that generates new instances, based on perturbations of instances from the original dataset, which are weighted by proximity to a instance of interest. Its initial implementation uses a linear model, but could in theory use any surrogate model. With the submodular pick algorithm (SP-LIME) [RSG16], the authors also proposed a global interpretability method, which picks a set of individual instances, that best represent the model's behaviour on a global level while at the same time restricting the number of instances to the minimum for selectiveness. [RSG16]

- *LOcal Rule-based Explanations (LORE)* [GMR+18a]: This is another method that generates synthetic instances around an instance of interest and trains a local model on this data. The local model consists of decision rules, that explain the reasons of a decision, and counterfactual rules. [GMR+18a]

2.4 SHAP

The interpretability method, that Mocca-SHAP is based on, is SHapley Additive exPlanations (SHAP) [LL17]. It has been proposed by Lundberg and Lee, and is an instance-level interpretability method. It unifies six existing methods [RSG16, SGK17, BBM+15, LC01, ŠK14, DSZ16] and is based on the game-theoretical concept of Shapley values [Sha53]. The resulting explanations are additive feature attributions. They are also models themselves with unique properties and can be used to base other explanations on. There are several extension methods, that allow global model interpretability: SHAP Summary Plots and SHAP Dependence Plots. [LL17, LEL19]

SHAP Values consist of a base value s_0 and a vector of additive feature attributions s, that sum up to the model's predicted value $\hat{y}$. The base value is the value, that would have been predicted, if no feature was present. [LL17, Mol20]

Regression Tasks

Example: A regression task with the three features x_1, x_2 and x_3, one numerical target variable y and a regression model $\hat{f}(x)$ to approximate the target variable with $\hat{y}$ is given. Assuming, that the feature attributions calculated by the SHAP algorithm for a specific instance are $s = (-5, 10, 0)^T$ and the base value $s_0 = 5$, then they together sum up to the model's predicted value of $\hat{y} = 10$. Further, they denote that x_3 has no effect on the target, because $s_3 = 0$. While x_1 has a decreasing effect, x_2 has an increasing effect on the target that exceeds x_1's effect. The base value explains, that if all features were absent, a value of 5 would be predicted.

Classification Tasks

SHAP Values only work with numeric outcomes. So for classification problems, one cannot explain the predicted labels directly. As discussed in Section 2.1, regression-based explainers can leverage probability estimates or their transformed alternative, log of odds estimates.

Example: Assuming, that the target variable of the previous example has a range of $(0, 10]$, then it can be discriminated into low values in the range $(0, 5]$ and high values in the range $(5, 10]$. The class for the lower range is termed 0 and that of the upper range 1. A probabilistic classifier is trained to predict these two classes. Its output is a probability vector, which has to sum up to 1. E.g. $\hat{y} = (0, 1)^T$ means, that the classifier assigns 100% to class 1 and 0% to class 0. The class with the highest probability in this case is class 1, which is actually predicted. For binary classifiers like in this example it seems trivial, because the probability of one class is the inverse of the other class, but for multiclass classifiers this distinction is required. The function to be explained is now not a single target variable anymore, but two target variables (or q variables for q target classes of the classification task). I assume that for the instance described in the previous example it calculates the SHAP Values of $s^0 = (0, -1, 0)^T$ and a base value of $s_0^0 = 1$ for

class 0; and another probability vector $s^1 = (0, 1, 0)$ and base value $s_0^1 = 1$ for class 1, which is the inverse probability. From this, I can interpret that with all features absent the classifier predicts class 0 with 100% and class 1 with 0%. Only feature x_2 has an effect on the outcomes, because its SHAP Values for classes 0 and 1 are non-zero. In this case it increases the predicted class 1 probability by 100% and decreases the class 0 probability by 100%. In this thesis, I combine all SHAP Value vectors into a matrix of dimension $p \times q$ and the base values into a vector of length q for p features and q classes.

2.4.1 Theoretical Background

The task of explaining a prediction can be seen as a cooperative game as in game theory, and therefore be explained using Shapley values [Sha53]. We can see the prediction task as a game, the features as players (called feature coalition), the prediction as the payout and the Shapley values as the distribution of the payout to the players. This is closely related to the weights in a linear regression model, where the solution to the distribution of the payout are the feature weights. Shapley values are contrastive, because they denote the change in outcome if a feature was absent. LIME [RSG16] tries to estimate the weights using a local regression model. With Shapley Values, this is the average marginal contribution of a feature value across all possible coalitions. The explanation model is defined as g for an original model f as in Equation 2.2, where $z' \in \{0, 1\}^M$ is the coalition vector, M the maximum coalition size, $s_j \in \mathbb{R}$ the feature attribution for feature j and s_0 the base value. [Mol20, Sha53]

$$g(z') = s_0 + \sum_{j=1}^{M} s_j z'_j \tag{2.2}$$

But most machine learning model implementations require all features to be present. Absence can be simulated by replacing a feature value with a random sample from the data set. This needs to be done for every possible feature coalition: (1) Estimate the outcome for a coalition. (2) Estimate the outcome where the feature of interest has been replaced by a randomly drawn value. The difference is the marginal contribution, which is then averaged across all possible feature coalitions. The sampling step can be done multiple times to get a better approximate. [Mol20, Sha53]

2.4.2 Special Properties

SHAP Values have special properties, that enable treating them as models and transforming and aggregating them. The properties are described differently by Lundberg and Lee than by Shapley [Mol20]:

- *Local Accuracy*: This property corresponds to the *efficiency* property of the Shapley value, where the sum of feature attributions has to match the difference of the prediction for an instance and the average prediction.

- *Missingness*: This property has not been described by Shapley [Sha53]. With this, Lundberg explicitly defines, that a constant feature has to have a SHAP Value of 0.

- *Consistency*: This property corresponds to the Shapley properties *linearity*, *dummy* and *symmetry*. These follow from consistency. It means, that *"if a model changes so that the marginal contribution of a feature value increases or stays the same (regardless of other features), the Shapley value also increases or stays the same."* [Mol20]. E.g. if in a retrained model of the example shown above feature x_2's effect increases, then $s_2 > 10$ has to be true. [LL17, Mol20, Sha53]

2.4.3 Unification of Interpretability Methods

Lundberg and Lee [LL17] argue, that their interpretability method unifies six existing approaches to local interpretability, listed below. [LL17]

- *LIME* [RSG16]: This method builds a local model, e.g. a linear model around an instance of interest, using a newly generated neighborhood and special weighting of the instances. This actually corresponds to the KernelSHAP algorithm with a different weighing function. The local linear model allows for interpretation of the weights as additive feature attributions. [LL17]

- *DeepLIFT* [SGK17]: This is a model-dependent explainer for deep neural networks, where SHAP's model-dependent optimized algorithm for neural networks is based on. It also explains by means of additive feature attributions.

- *Layer-Wise Relevance Propagation* [BBM+15]: This method is another model-dependent explainer for deep neural networks, similar to DeepLIFT. [LL17]

- *Shapley regression values* [LC01]: Shapley Values are based on the idea of measuring the effect on the outcome when a feature is absent. With this algorithm, the model is retrained with all possible combinations of missing features. Apart from being very inefficient for complex models, that require much time for training, it is not always possible to retrain the model. In post-hoc interpretability tasks one often only has access to a trained model and data. [LL17]

- *Shapley sampling values* [ŠK14]: In contrast to Shapley regression values, this method approximates the effect of a feature being absent by replacing its value with samples from other instances and then taking the average. [LL17]

- *Quantitative input influence* [DSZ16]: This is a broader framework, in which feature attributions are calculated identically to Shapley sampling values, but has been independently proposed. [LL17]

2.4.4 Algorithms to Compute SHAP Values

The simplest model-agnostic approach was described by Shapley [Sha53], and has been refined by Štrumbelj and Kononenko [ŠK14], now termed Shapley sampling values. Its time complexity is $O(M2^M)$, because it completely enumerates the space of masking patterns. A slightly more efficient implementation can be found in the SHAP python package[2] with the `shap.explainers.Exact` explainer [Lun21]. [LL17]

Lundberg and Lee prove, that SHAP Values can be computed by solving a linear regression problem using a special weighing kernel. They term this algorithm *Kernel SHAP* [LL17]. They note, that it has better sample efficiency than using the default approach described in the previous paragraph. LIME computes feature attributions in a similar manner, but by using a different weighing kernel, one based on proximity. [LL17]

Further model-dependent algorithms have been proposed, that make use of extra knowledge about the model to speed up computation. First, there is *Deep SHAP* [LL17] for deep neural networks, which works similarly like DeepLIFT [SGK17], but has been adapted to yield feature attributions that satisfy SHAP's special properties. And second, there is *Tree SHAP* [LEL19] for decision trees and tree-based ensemble models. [LL17, LEL19]

For this thesis, I limit my selection to model-agnostic algorithms, to be able to demonstrate my proposed method for explaining differences between models in a consistent, general way. Of these, I choose the original algorithm described by Shapley [Sha53] as implemented in the SHAP Python package. Kernel SHAP would have been a valid choice as well, but either of the two suffices. In practical scenarios, one might choose one of the faster model-dependent algorithms, if deep neural networks or tree-based models are to be explained.

2.4.5 Global Explanations

When SHAP Values are calculated for an entire data set, they can be combined to explain a model's behaviour on the global level. They can be aggregated to obtain feature importances or plot for a feature like is done with SHAP Summary Plots and SHAP Dependence Plots. In the following sections, I explain them in more detail.

Until now, I have described how SHAP can be used to explain a single instance. But by combining many instances' explanations, a model's behaviour can be explained on the global level. Lundberg, Erion and Lee propose SHAP Summary Plots and SHAP Dependence Plots as global level extension methods [LEL19], which I describe in the following sections.

SHAP Feature Importance

Considering, that large absolute values imply importance, one can calculate the average magnitude of the SHAP Values per feature as in Equation 2.3. s denotes the SHAP

[2] Python package shap: `https://github.com/slundberg/shap`

Values, p the number of features, q the number of classes and n the number of instances. As implemented in the SHAP Python package, feature importances are visualized in a bar chart, with features sorted in descending order by their importance. This way, the most important features can be spotted fast. See Figure 2.1 for an example, where the features *Relationship* and *Age* are of highest importance. The disadvantage of this statistic is, that details are hidden in the aggregated values. [LEC+20]

$$\frac{\sum_{k=1}^{q}\sum_{i=1}^{n}|s_{i,j,k}|}{nq}, \forall j = 1, \ldots, p \tag{2.3}$$

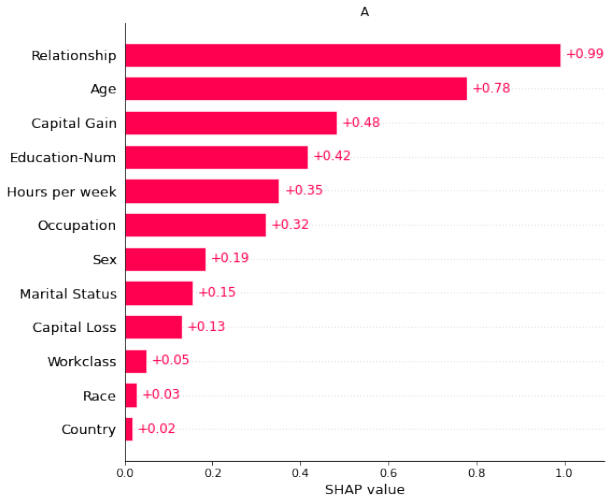


Figure 2.1: SHAP Feature Importances of classifier A for the positive class' log odds estimates of the Adult (Census Income) example. Features are sorted in descending order by their importance.

SHAP Summary Plots

SHAP Summary Plots also allow to interpret feature importances, but in more detail. A chart actually shows the SHAP Value distribution per feature, with features ordered by their importance. They were originally proposed as an alternative to traditional violin plots, which are also used to visualize distributions, but they do not rely on a smoothing kernel and just stack scattered points with similar values. This way, dots pile up in dense areas of the distribution. In contrast to SHAP Feature Importances, one can detect groups of instances by a feature's effect. For example, Figure 2.2 shows that the most important feature *Relationship* either has a negative or a positive effect on the outcome, with no cases in between. The authors further propose to color the instances by the actual feature value, which allows a rough interpretation about effects of different feature values. In the example, I interpret that low values of the second most important feature *Age* have a decreasing effect on the outcome, while high values have an increasing effect, with a gradual shift in between. [LEL19]

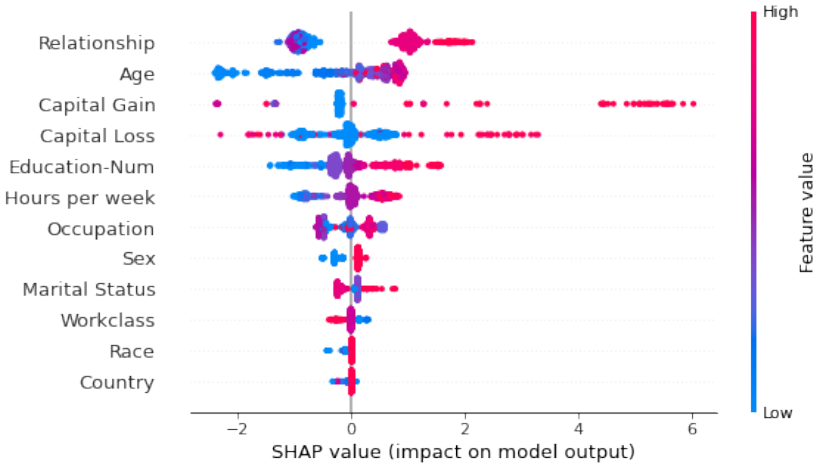


Figure 2.2: SHAP Summary Plot of classifier A for the positive class' log odds estimates of the Adult (Census Income) example. Features are sorted in descending order by their importance.

SHAP Dependence Plots

SHAP Dependence Plots make interpretation of the type of relation between feature and effect on the outcome easier than with SHAP Summary Plots. Instead of encoding the actual feature value of each instance in the color domain, it is visualized as a separate axis. A SHAP Dependence Plot is created for one feature at a time, depicting its feature values on the x-axis vs. its SHAP Values on the y-axis in a scatter plot with one dot per instance. There is a close connection to Partial Dependence (PD)-plots [Fri01]. With both SHAP Dependence Plots and PD-plots one can draw conclusions about how an outcome changes if a feature changes. [LEL19]

In the following paragraphs, I present some examples to compare interpretation with PD-plot and SHAP Dependence Plots.

Constant Relation: In Figure 2.3 you can see a feature that has no effect on the outcome. In this case, the dependence curve is constant. Both methods perfectly describe this behaviour: SHAP Dependence Plots with scatter points, PD with a line. But the y axes of the plots are different: For SHAP Dependence Plots, it is constantly zero, which means that this feature neither increases nor decreases the outcome for every instance. With PD-plots, it is constantly 0.5, which happens to be the decision boundary value for probability estimates of this binary classifier. The close relation between the two is, that SHAP Dependence Plots in this case resemble a mean centered version of the PD-plots.

Linear Increase/Decrease: There may be a linear relation between a feature and an outcome, as shown in Figure 2.4. It can be either increasing or decreasing.

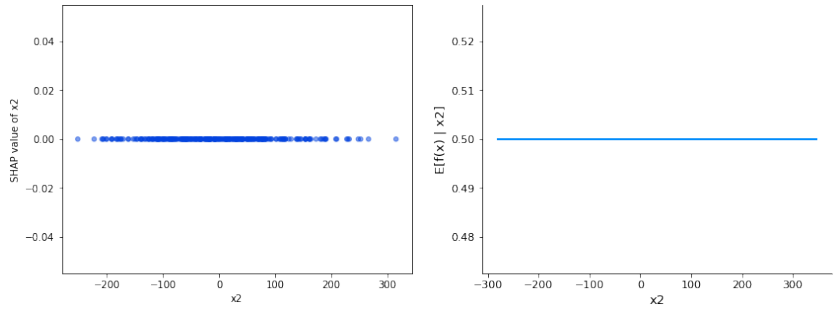


Figure 2.3: SHAP Dependence Plot (left) and PD-plot (right) of classifier A's positive class probability estimates vs. x_2, running example, described in Section 4.1. Both perfectly describe the constant shape of the dependence curve.

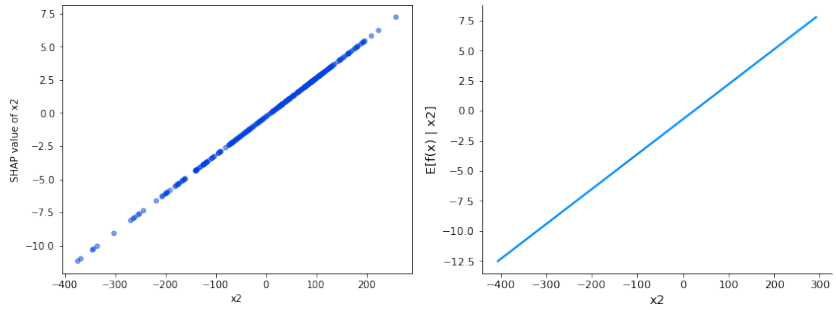


Figure 2.4: SHAP Dependence Plot and PD-plot of classifier A's positive class log odds estimates vs. x_2, "One Classifier Ignores a Feature" example, described in Section 4.2. Both perfectly describe the linear increasing relation.

Monotonic Increase/Decrease: A more general way is to describe the relation as monotonically increasing or decreasing, without assuming linearity. See Figure 2.5 for an example

Concave/Convex: When the dependence curve forms a low after decreasing monotonically or a high after increasing monotonically and then reverses direction, it is called concave and convex respectively. See Figure 2.6 for an example.

Step Up/Down: The dependence curve may also be of a step-like nature, as shown in Figure 2.7.

Other Nonlinear Relation: In real-world settings, a dependence curve can be any arbitrary shape. Most of the time, it is possible to describe parts of the curve with the previous terms.

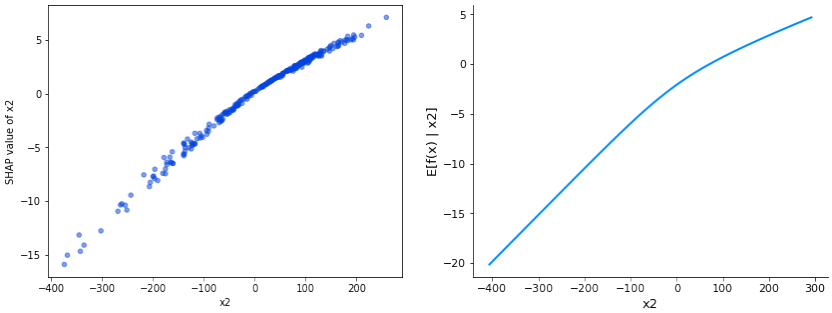


Figure 2.5: SHAP Dependence Plot and PD-plot of difference class $(1, 1)$'s log odds estimates vs. x_2, "One Classifier Ignores a Feature" example, described in Section 4.2. Both show a monotonically increasing curve.

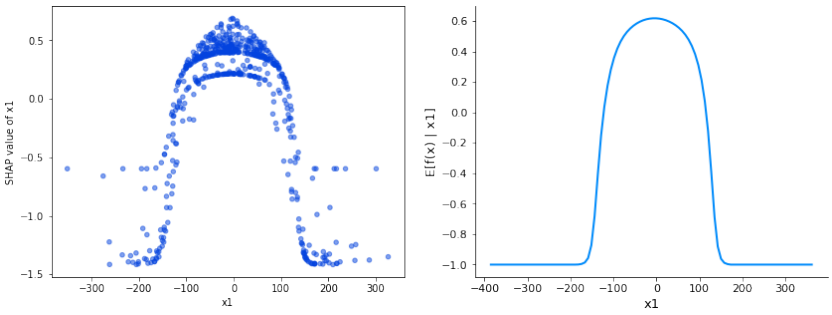


Figure 2.6: SHAP Dependence Plot and PD-plot of classifier B's negative class log odds estimates vs. x_1, Gaussian Quantiles example, described in Section 4.3. Both approaches show the convex shape of the dependence curve, which is increasing at first, and decreases again around 0, forming a global high. The SHAP Dependence Plot additionally shows vertical spread, which is especially big on the lower and upper end. This is caused by interaction effects.

A disadvantage of PD-plots is, that the resulting curve only shows the average effect. But because of interactions with other features, there may be different relations present. SHAP Dependence Plots show variance on the y-axis, and with interactions, instances are much more dispersed. By visualizing another feature's values in the color dimension, it enables interpretation of feature interactions. See Figure 2.8 for an example. [LEL19, Mol20]

Model Output Spaces: But which output space is suited best for interpretation? Lundberg uses log of odds in his works for classification tasks [LL17, LEL19, LEC+20]. But he further notes, that this depends on the type of the model [Lun]. If probability estimates are available and they are mostly soft probability estimates, then the log of odds output space is preferable. Soft probability estimates are in contrast to hard probability estimates not only either 0% or 100% for a class, but mostly between these extremes.

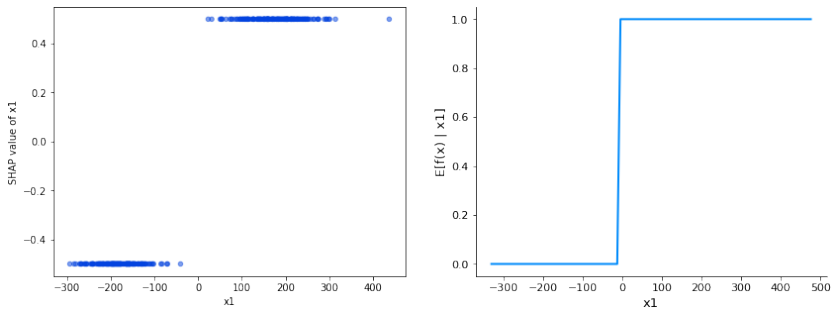


Figure 2.7: SHAP Dependence Plot and PD-plot of classifier A's positive class probability estimates vs. x_1, running example, described in Section 4.1. Both approaches show the step up at approx. 0, below and above it is constant.

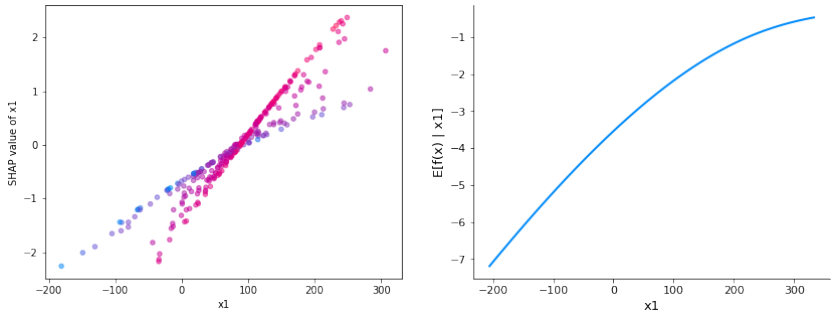


Figure 2.8: SHAP Dependence Plot and PD-plot of difference class $(0, 1)$'s log odds estimates vs. x_1, "One Classifier Ignores a Feature" example, described in Section 4.2. In the SHAP Dependence Plot, the instances are colored by their x_2 value from blue to red for increasing x_2 values, which allows one to see, that the dependence curve is steeper for higher x_2 values than for lower values. The PD-plot is misleading as it just shows the average effect.

In Figures 2.9, 2.10 and 2.11 you can see the SHAP Dependence Plots of feature x_2 and the various model output spaces for classifier A of task "One Classifier Ignores A Feature" (described in Section 4.2), a logistic regression classifier. The simplest feature-effect relation is the one in the log of odds output space, because it is linear. The other output spaces result in a more complex relation, which is shaped sigmoid and includes vertical spread. The vertical spread accounts to feature interaction. The only drawback of using log of odds space is, that values are not as intuitive as interpreting changes in percent. This drawback is outweighed by the fact, that log of odds values are naturally additive. In all my experiments, I use the log of odds space.

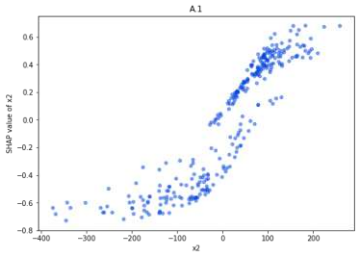


Figure 2.9: one-hot encoded class label SHAP Values

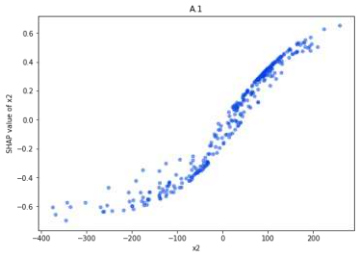


Figure 2.10: Probability SHAP Values

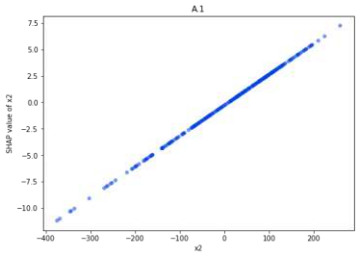


Figure 2.11: Log of Odds SHAP Values

Clustering

Lundberg and Lee propose a new method for clustering, termed *supervised clustering* [LEL19]. This should not be confused with the term "supervised" as in the sense that "supervised machine learning" works with known true labels. In fact, the true labels are not required by any SHAP method at all. I interpret, that the authors wanted to highlight, that this form of clustering is not only based on the features, but also on the predictions of the model. This clustering method applies a hierarchical clustering algorithm on the explanations themselves, the SHAP Values. They state, that this solves one of the most challenging problems in clustering: The problem of how to assign feature weights, which is equivalent to the question about how to determine a distance matrix. SHAP Values are always in the unit of the outcome space, which is the same for every feature, thus there is no need for scaling or determining weights. The authors further note, that hierarchical clustering encodes many possible groupings of instances within the hierarchical structure. The previously presented global extension methods can also be applied to a group of instances, which allows one to get local explanations in a modular way. [LEL19]

2.5 Traditional Approaches to Model Comparison

A lot of research has been conducted on interpretability methods that seek to explain single models, as can be seen in the collections in the books of Molnar [MCB20] and Biecek [BB21]. Yet there has not been done much research on comparing models using interpretability. In most works, this is done by visualizing the explanations for the models side-by-side or plotting them into the same chart. These are described now in the following sections.

2.5.1 DALEX R package

Biecek states in the paper of his R package `DALEX` [Bie18], standing for Descriptive mAchine Learning EXplanations, that all described explainers can be natively used to compare models. He suggests the use of side-by-side bar charts for feature importances,

obtained with the method proposed by Fisher, Rudin and Dominici [FRD19] (see Figure 2.12). He further suggests to visualize PD profiles in a single chart (see Figure 2.13). For comparing local explanations he suggests to use side-by-side charts that show SHAP Values (see Figure 2.14). [Bie18]

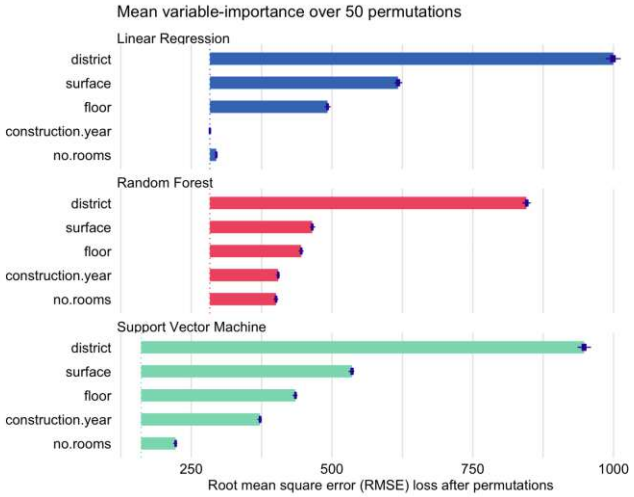


Figure 2.12: Feature importances for three different models [BB21, p. 200ff.].

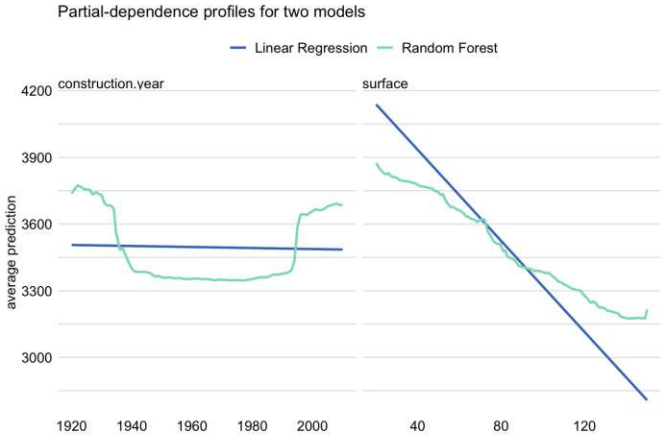


Figure 2.13: Partial dependence profiles for two different models, each chart shows one feature [BB21, p. 217].

2.5.2 Prospector analytics system

Krause et al. proposed an interactive visual analytics system, called *Prospector* [KPN16]. It uses PD-plots in the same way how Biecek uses them to compare feature effects for

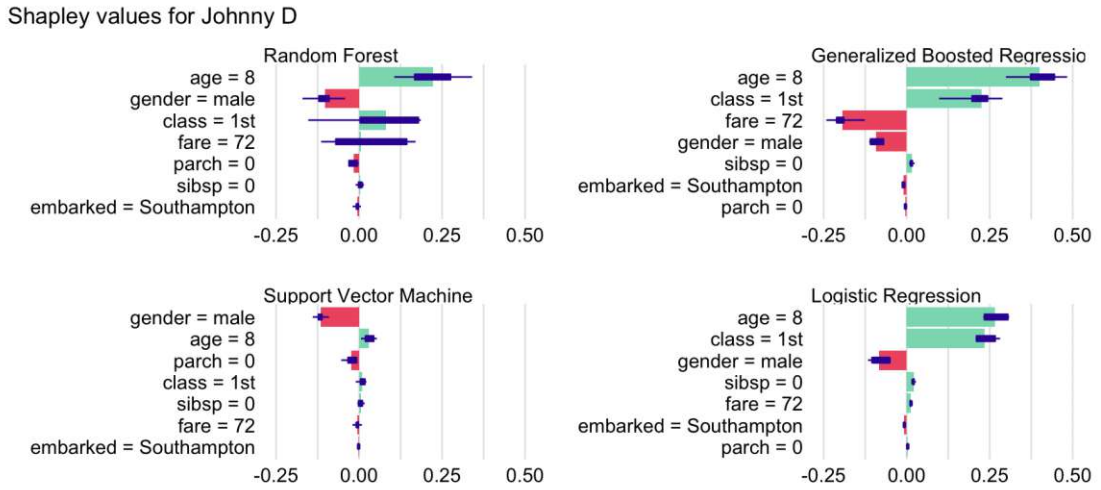


Figure 2.14: SHAP values of a single instance for four different models [BB21, p. 164ff.].

multiple models. But Prospector adds features for interactive instance-level interpretation, which is required to be done manually with Biecek's tools. [KPN16]

2.5.3 Manifold framework

Zhang et al. proposed another framework for interpretation of models, called *Manifold* [ZWM+18]. It combines a model performance comparison graph and feature importance comparison graph in one dashboard. [ZWM+18]

2.5.4 explAIner framework

The proposed visual analytics framework *explAIner* of Spinner et al. [SSSEA19] is tightly integrated with the iterative process of model building and provides single model explainers as well as multi-model explainers. Different explainers can be used, but most of the listed ones only support deep learning models. For example, the black-box explainers LIME [RSG16] and ANCHORS [RSG18] are supported. Yet, the authors do not explicitly describe, how comparison works. [SSSEA19]

2.6 DiRo2C

The instance-level interpretability method Difference Recognition of 2 Classifiers (DiRo2C) [Sta21] takes a different approach. Instead of visualizing explanations side-by-side or two explanations in the same chart, Staufer and Rauber propose to explain differences directly by reformulating the explanation task via a difference classification task. It applies an adapted variant of the local interpretability method LORE [GMR+18a] to explain an instance with the difference classifier. This variant uses a genetic neighborhood generation

algorithm to create synthetic instances with which a local decision tree surrogate model is trained, which in turn is used to derive a decision rule set. The final explanation is a description of local differences. Further, an extension was proposed that allows to explain globally, by iterating over instances of a sample of the data set, generating the neighborhood for each and merging them into a single data set before training a surrogate decision tree and deriving rules. It uses a 10% sample in the reference implementation. Because the explanation itself is a decision tree surrogate model, further explanations can be derived, e.g. feature importances from model internals. [Sta21]

2.6.1 Difference Classifier

In this section I describe the difference classifier, as used by Staufer and Rauber in DiRo2C [Sta21] and investigate possible variations and extensions for probability estimates to tackle research question #2.

DiRo2C is based on an abstraction of the problem of applying an interpretability method to compare models via the use of an intermediate classification problem. The so-called *difference classifier* is explained by a normal interpretability method. It is not an actual classifier that needs to be trained, and thus is not an approximation. Instead, it is a merger of the outputs of the individual classifiers. The difference classification problem can be formulated in different ways, each including different classes. Staufer and Rauber proposed two forms: (1) the binary difference classifier and (2) the multiclass difference classifier. I also look into possible other variations and examine them in the next sections. [Sta21]

Binary Difference Classifier

In the simplest form, the *binary difference classifier*, it tells whether the individual classifiers predict equal labels or different labels. Equal labels are predicted as class 0 (negative label) and different labels as class 1 (positive label). It can be described using a simple decision rule, where $\hat{y}$ is the prediction of a classifier, as shown below. In this way, it works for classifiers with any number of classes.

$$\text{if } \hat{y}_A = \hat{y}_B, \text{ then label 0, else 1}$$

Multiclass difference classifier

The multiclass difference classifier predicts a separate class per type of difference or equality. I start with the definition of the simpler case, where two binary classifiers are compared. Then it has the two difference classes $(0, 1)$ and $(1, 0)$ and the two equality classes $(0, 0)$ and $(1, 1)$:

- if $\hat{y}_A = 0 \wedge \hat{y}_B = 0$, then label 0 (called $(0, 0)$)
- if $\hat{y}_A = 0 \wedge \hat{y}_B = 1$, then label 1 (called $(0, 1)$)

- if $\hat{y}_A = 1 \wedge \hat{y}_B = 0$, then label 2 (called $(1, 0)$)
- if $\hat{y}_A = 1 \wedge \hat{y}_B = 1$, then label 3 (called $(1, 1)$)

The general definition for merging the outputs of multiclass difference classifiers is shown below. This problem formulation includes m^2 classes, where m is the number of classes of the individual classifiers. It includes m equality classes and $m^2 - m$ difference classes. This can be seen as the classes in the confusion matrix of the predictions of classifier A vs. classifier B.

- if $\hat{y}_A = 0 \wedge \hat{y}_B = 0$, then label 0 (called $(0, 0)$)
- if $\hat{y}_A = 0 \wedge \hat{y}_B = 1$, then label 1 (called $(0, 1)$)
- if $\hat{y}_A = 0 \wedge \hat{y}_B = 2$, then label 2 (called $(0, 2)$)
- ...
- if $\hat{y}_A = 0 \wedge \hat{y}_B = m - 1$, then label $m - 1$ (called $(0, m - 1)$)
- if $\hat{y}_A = 0 \wedge \hat{y}_B = m$, then label m (called $(0, m)$)
- ...
- if $\hat{y}_A = 1 \wedge \hat{y}_B = m$, then label $2m$ (called $(1, m)$)
- ...
- if $\hat{y}_A = m \wedge \hat{y}_B = m$, then label m^2 (called (m, m))

2.7 Summary

In this section, I have described that this work focuses on classifiers. But because of the nature of the chosen interpretability method SHAP it is necessary to treat them as regression models. Two output spaces of classifiers can be used for explanations: probability and log of odds space. I have defined the terms interpretability and explanations. Explanations are contextual, which includes that they should be contrastive, selective, social and that probabilities are not as important as causations [Mil19]. To tackle the problem that this thesis addresses, model-agnostic post hoc interpretability methods are required. Yet in the case of surrogate models, intrinsic interpretability and model-specific extension methods may be used as well. I have described both global and local level understanding. I have given an overview about the explanation types feature summary statistics, data points (including counterfactual explanations), model internals and surrogate models. I described SHAP [LL17] and how the special properties allow to derive feature importance measures and aggregate local explanations to explain globally with SHAP Summary Plots and SHAP Dependence Plots [LEL19]. I have

further described algorithms to compute them and how they can be used in conjunction with classification tasks and that they are suited well for clustering. I have also given an overview about traditional model comparison approaches using interpretability and described the more targeted approach of DiRo2C [Sta21] with the difference classifier.

CHAPTER 3

Research Methodology

In this work I use Design Science by Hevner et al. [HMPR04] as the scientific method to iteratively develop and evaluate the artifact Mocca-SHAP. The strategies described in the evaluation framework by Pries-Heje et al. [PHBV08] assist during evaluation. In Section 3.1, I describe how I fit my research into this framework. In Section 3.2, I review literature regarding interpretability and related work, that targets model comparison with interpretability. In Section 3.3, I describe how I select data sets, set up the experiments and evaluate.

3.1 Design Science

Design Science [HMPR04] is one of the foundational paradigms of the information systems discipline. Its goal is to create new and innovative artifacts, rather than describing or predicting behaviour. It offers a conceptual framework and research guidelines. In the following sections, I describe how I fit my research into this framework and how the research guidelines are taken into account. [HMPR04]

3.1.1 Research Cycle

Design Science research can be seen from the Three Cycle View [Hev07] and the three pillars that the closely related cycles connect (shown in Figure 3.1):

- *Environment*: People and organizations in my environment includes data scientists and their stakeholders. Data scientists have a need which the artifact should satisfy, termed Relevance Cycle. Their stakeholders are their employer, their customers and the lawmaker. Employer and customer require the development process to be transparent and the final selection to yield optimal and unbiased results. Further, they are required to comply with the GDPR [Eur16]. [HMPR04]

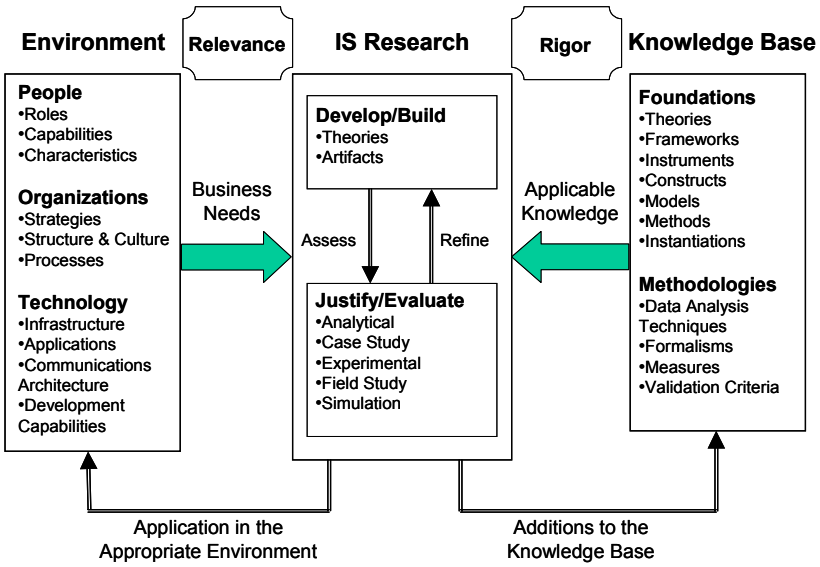


Figure 3.1: The Design Science research cycle [HMPR04].

- *Knowledge base*: I am building on the previous research about interpretability evaluation [PHBV08, DVK17], model comparison [Bie18, Fri01, ZWM+18, SSSEA19, Sta21], and all the previous research on model interpretability, especially SHAP [LL17]. With the artifact, I can contribute to the research about model comparison, uses of SHAP in new ways and a prototypical implementation that may be used by anyone free to use or extend. This is called the Rigor Cycle. [HMPR04]

- *Information systems research*: With the previous two points providing interacting context to my work, this is the main part where I develop the artifact iteratively and evaluate it in comparison to other methods. This cycle of assessment and refinement is called the Design Cycle. [HMPR04]

3.2 Literature Review

Via an initial recommendation by my advisor of the book *"Interpretable Machine Learning"* by Molnar [Mol20] I was able to get an overview about interpretability methods. With discussions about the notion of the difference classifier, initially proposed by Staufer and Rauber [Sta21], I was able to find a state-of-the art approach about model comparison with interpretability. Via a reference in the book of Molnar, I discovered another book on the topic: *"Explanatory Model Analysis"* by Biecek and Burzykowski [BB21]. It too was of great help at understanding and categorizing interpretability methods and further showed examples of comparing models with interpretability. I followed the cited literature in the books and complemented that by using the Google Scholar search engine with common terms like *interpretability* and *explainability* in conjunction with

machine learning to discover evaluation strategies and surveys regarding interpretability methods. The selected results are listed below. Further, I used the term *model comparison* additionally to discover other approaches, which are described in Section 2.5.

- Surveys about interpretability methods:
  - *"Machine Learning Interpretability: A Survey on Methods and Metrics"* by Carvalho, Pereira and Cardoso [CPC19]
  - *"Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)"* by Adadi and Berrada [AB18]
  - *"A Survey of Methods for Explaining Black Box Models"* by Guidotti et al. [GMR+18b]
  - *"Perturbation-Based Explanations of Prediction Models"* by Robnik and Bohanec [RŠB18]
- Literature about interpretability method evaluation:
  - *"Towards A Rigorous Science of Interpretable Machine Learning"* by Doshi-Velez and Kim [DVK17]
  - *"Explanation in Artificial Intelligence: Insights from the Social Sciences"* by Miller [Mil19]
  - *"Evaluating Explanation Without Ground Truth in Interpretable Machine Learning"* by Yang, Du and Hu [YDH19]
  - *"Explainability Fact Sheets: A Framework for Systematic Assessment of Explainable Approaches"* by Sokol and Flach [SF20]

3.3 Experiments

According to Doshi-Velez and Kim, *"core methods work should demonstrate generalizability via careful evaluation on a variety of synthetic and standard benchmarks"* [DVK17] in machine learning model interpretability. Iteratively, I develop a method and evaluate in comparison to a baseline and State-of-the-art (SOA) method. Qualitative and quantitative quality metrics assist during comparison. Evaluation is done in an application-grounded manner in the context of the end task, which is noted by Doshi-Velez and Kim to be the preferred way for evaluating an interpretability method for a concrete application [DVK17]. At first in a lab setting, then in a benchmark setting using real world datasets as suggested in the evaluation framework by Pries-Heje et al [PHBV08].

In this section, I describe the selection of data sets, the types of differences that can be introduced into the classifiers and finally evaluation with regards to a selection of quality metrics.

3.3.1 Data Sets

In this work I use tabular data sets, although extensions with support for text and image exist for SHAP. But this would be out of scope. I do not restrict the feature types, they can be any type, including categorical and numerical types. I will use artificial data sets in binary classification tasks to demonstrate how the method works. I will use two data sets published by Staufer and Rauber [SR21b, SR21a] and develop further ones as required in a similar fashion. To demonstrate in a benchmark setting, I will use real-world data sets. The first one is the Adult data set, which is also known as Census Income data set and was used by Staufer and Rauber [Sta21], available in the UCI Machine Learning Repository [DG17]. Its target is a binary label. To evaluate the methods on a multiclass-classification task as well, I choose the Boston Housing data set [HJR78], which was recommended by my advisor along with a difference recognition task.

3.3.2 Controlled Experiments

To be able to do a controlled experiment, I need to know before training, what the ground truth of differences is that are to be explained. There are different options used throughout this work, all based on artificial difference introduction methods. In the simpler examples inherently interpretable models are used, and therefore an exact description of their differences can be derived by looking at their model internals. But for more complex models, this is not possible anymore. Then it is necessary to rely on knowledge about the differences and check, in which ways the classifiers have picked them up. Non-artificial difference introduction methods include comparing classifiers with different inherently known characteristics and classifiers trained on different parts of a data set, which is known to have a concept drift. But to demonstrate basic efficacy of the artifact, the artificial methods listed below are sufficient:

- *Synthetic training data set generation:* If the training data sets are generated synthetically, it is possible to generate different distributions and therefore influence how classifiers work internally.
- *Set a feature to a constant value:* Setting a feature in the training data set to a constant value makes it effectively unusable for a classifier. Therefore, it needs to rely on other features for making decisions.
- *Transform features with mathematical operations:* In the simplest case, a constant value is added to or subtracted from each feature value. Other transformations are also possible.
- *Change labels with rule-based logic:* The true labels used to train a classifier can also be modified. Using a rule-based logic, certain instances are selected for which labels are set to a specific value.

3.3.3 Evaluation

The artifact is evaluated ex post in the context of the application with experimental designs and a domain expert, according to the design science evaluation strategies suggested by Pries-Heje et al. [PHBV08]. Ex post means, that the artifact is implemented first and then evaluated. Experimental designs allow automatic computation of quality measures, which are described in Section 3.3.3. These experimental designs are run in a laboratory setting with controlled parameters. In the evaluation framework of Doshi-Velez and Kim [DVK17], this serves a function level evaluation, which is appropriate, when the interpretability method has been evaluated on the human level before, as is the case with SHAP [LL17]. Since they are applied in a different way and extended, they need to be evaluated in user opinion studies again. But because of the resource constraints of this master thesis, which may include time and access to domain experts [VPHB12], I am doing this in a descriptive way with myself as a domain expert only. As Hevner [HMPR04] notes, this is appropriate if the artifact is especially innovative and other forms are not feasible. However, a follow-up user study is necessary to fully test the hypotheses with a broader audience, and this thesis can only indicate if the developed artifact is a promising approach. [DVK17, HMPR04, PHBV08, VPHB12]

To answer research question #1, a variety of quality metrics is selected that are then used to evaluate the developed artifact in comparison to the SOA and a baseline. Some are quantifiable, some need qualitative description. Molnar [Mol20] writes in his book about evaluation metrics and properties of explanations. This is mostly based on the works of Doshi-Velez and Kim [DVK17], Robnik and Bohanec [RŠB18] and on the survey of Miller [Mil19]. Adadi and Berrada further mention in their survey, that Miller's is the most significant attempt at linking human science and XAI. Carvalho, Pereira and Cardoso [CPC19] describe qualitative metrics that impact comprehensibility in more detail. [AB18, CPC19, Mol20]

Fidelity measures, how precisely the explanation is able to approximate the model. Generally, this answers the question: "Can I correctly predict the model's outcomes?". It can be measured with the same techniques, that the performance of a classifier is measured. These include Precision, Recall, Accuracy and F1 score but is calculated based on the predictions of the explanation and the predictions of the classifier to be explained. Some explanations like surrogate models directly allow to predict instances, while other explanation types may require a proxy. [RŠB18, Mol20]

Complexity measures, how many cognitive chunks an explanation contains. A cognitive chunk refers to the basic unit an explanation is made of. For example, surrogate decision rules can be measured by their total number of constraints. This quantifiable metric can be used to qualitatively evaluate selectiveness, or sometimes called sparsity. Humans want explanations to be selective, thus to focus only on the most important aspects, and not to be complete. [DVK17, Mol20]

Generation time measures, how long explanation generation actually takes on a target machine. This is expected to resemble the algorithmic complexity, which can only be

compared in general for the methods. [RŠB18]

By taking into account these quantifiable metrics, I will evaluate qualitatively how well the explanations satisfy Miller's desirable properties of explanations **contrastiveness**, **selectiveness**, **causality** and **interactivity**, together making up contextuality [Mil19]. They are described in Section 2.2.1.

Further properties have been proposed to be desirable for explanations, but with those mentioned above I have put together a selection that covers the most important aspects, while being feasible to evaluate within the limits of this thesis. For example, Robnik and Bohanec [RŠB18] define the properties of explanations with *expressive power*, *translucency*, *portability* and *algorithmic complexity*. Portability will be ensured during development and algorithmic complexity described with complexity. The authors further define the quality of explanations with accuracy, fidelity, consistency, stability, comprehensiblity, certainty, degree of importance, novelty and representativeness. A part of these will be covered with the selected metrics. Carvalho, Pereira and Cardoso [CPC19] define the quality metrics in a different way with *form of cognitive chunks*, *number of cognitive chunks*, *compositionality*, *monotonicity and other interactions* and *uncertainty and stochasticity*. This is yet another view on the same properties. [RŠB18, CPC19]

3.4 Summary

I am using the methodology of Design Science [HMPR04] when building the method Mocca-SHAP to ensure relevance, which means that the needs of data scientists and their stakeholders are met, and rigor, which means that I build on previous scientific work about interpretability methods and related work about model comparison and add to this knowledge base with my research. Building the artifact involves a cycle of refinement and assessment, for which I selected two artificial data sets and two real-world data sets. I will create further ones if necessary. Evaluation is done in controlled experiments, for which I have put together artificial difference introduction methods. The quantifiable metrics *fidelity*, *complexity* and *generation time* are automatically derived for each task. Taking these into account, I will qualitatively evaluate Mocca-SHAP's explanations with regards to Miller's desirable properties [Mil19] *contrastiveness*, *selectiveness*, *causality* and *interactivity* in comparison to the SOA approach and a baseline. This answers research question #1. However, a follow-up user opinion study is required to fully evaluate the artifact.

CHAPTER 4

Difference Recognition Tasks

In this section, I describe the difference recognition tasks on which Mocca-SHAP will be evaluated in comparison to DiRo2C and a baseline. Three tasks are based on artificial data sets with two features and two tasks are based on benchmark data sets with more than two features.

- In the *Running* example, the task is to compare two simple decision trees that have three distinct parts in the feature space where they predict differently.
- In the *One Classifier Ignores a Feature* example, the task is to compare two logistic regression classifiers, where one of them is only able to make use of one of the two features during training. Two parts in the feature space are predicted differently, but their description involves feature-interdependent rules.
- In the *Gaussian Quantiles* example, the task is to compare two SVCs. One part in the feature space is predicted differently. It cannot be described analytically anymore, but is in the shape similar to a ring.
- In the *Adult* example, the task is to compare two ensemble classifiers trained on the well-known Census Income data set, also known as the Adult data set. One classifier is trained with a data set were one feature has a constant value added to it, which results in it predicting the negative label more often than the other classifier on the unmodified data.
- In the *Boston Housing* example, the task is to compare three MLP classifiers, that have been trained on the Boston Housing data set where the goal is to predict a three-class target.

4.1 Running Example

This simple example is to show how Mocca-SHAP, DiRo2C and the baseline approach basically work. The data set and the task were originally published by Staufer and Rauber [SR21b]. The data set is generated synthetically, has a small size with 300 instances and two numerical features and is easily visualizable in a two dimensional scatter plot. Two separate groups are present in the data, one associated with class 0 and the other with class 1, making it a binary classification task. Both groups' instances are sampled from a normal distribution and transformed differently. They are linearly separable by feature x_1 without causing misclassifications. 80% of the data set are used for training and 20% for evaluating classifier performance. The classifiers to be compared are decision trees. Classifier A is trained on the original version and achieves perfect accuracy. Two manipulations are applied to the data set used to train and evaluate classifier B:

1. if $x_1 < 150 \wedge x_1 > 0 \wedge x_2 > -100 \wedge x_2 < 100$, then change label to 0
2. if $x_1 < 0 \wedge x_1 > -200 \wedge x_2 \geq 100$, then change label to 1

Classifier B also achieves perfect accuracy on its held out test set. As the data set to generate explanations for, A's test set will be used. Both classifiers are simple enough to be considered interpretable with A having three and B having eleven decision nodes. A plot of their decision boundaries is shown in Figure 4.1. The different inner workings result in three areas being classified differently, as can be seen in the decision boundaries of the difference classifier in Figure 4.2. The difference classifier predicts one instance as $(0, 1)$ and eight instances as $(1, 0)$. Although the classifiers offer probability estimates, there is no advantage in using them. Both classifiers only predict either 0% or 100% for a class, which makes the probability estimates equal to the one-hot encoded labels.

4.1.1 Ground Truth

Because small decision trees are inherently interpretable, I can comprehend how they decide internally by looking at their inner structure. Figure 4.3 shows their decision nodes. From this, I manually derive a decision rule set that perfectly describes the differences. The rules are listed below. Values have been rounded to one decimal for display.

1. if $-203.8 < x_1 \leq -8.5 \wedge x_2 > 97.5$, then $(0, 1)$
2. if $-16.4 < x_1 \leq -8.5 \wedge x_2 \leq -108.2$, then $(0, 1)$
3. if $-8.5 < x_1 \leq 150.7 \wedge -108.2 < x_2 \leq 97.5$, then $(1, 0)$
4. else, the classifiers predict equal labels

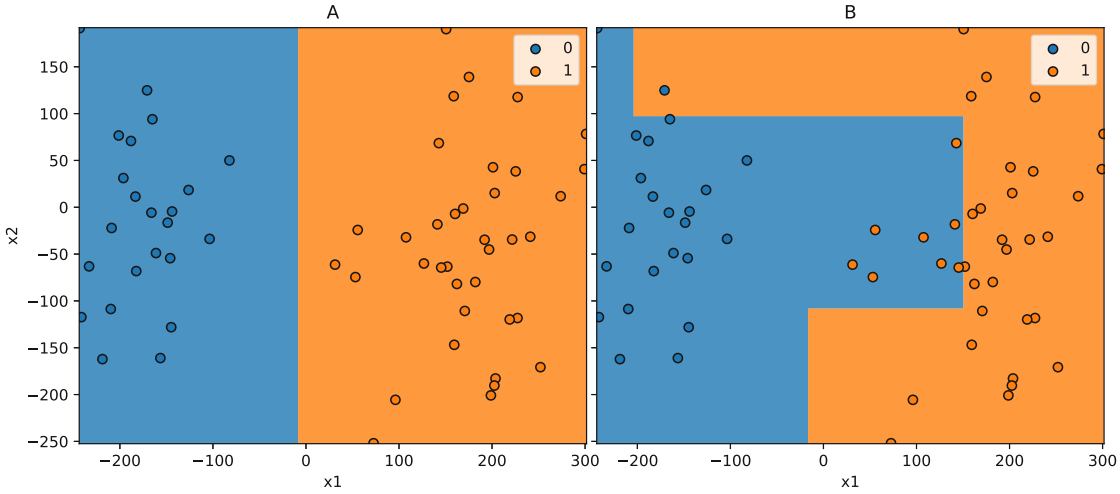


Figure 4.1: Each classifier's decision boundaries, overlaid by a scatter plot of the instances to be explained of the running example. The color denotes the true label as present in the original data set.

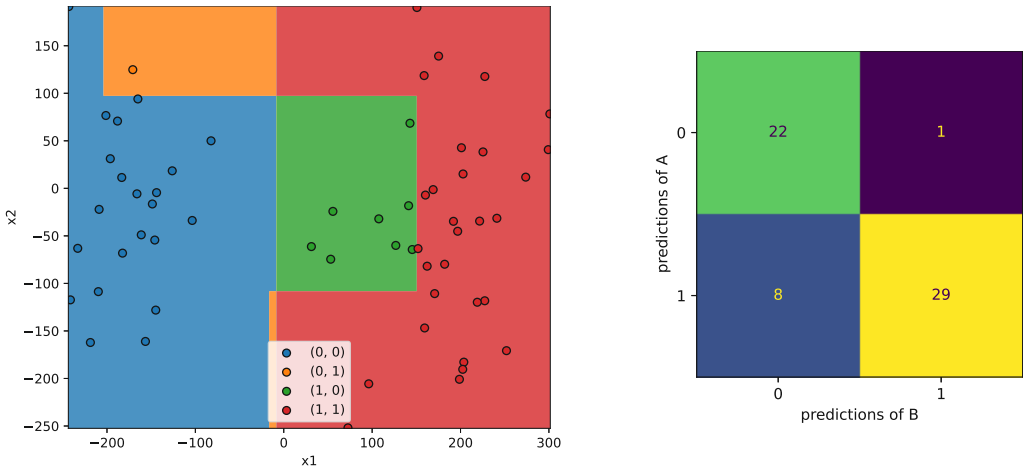


Figure 4.2: Difference classifier decision boundaries, overlaid by a scatter plot of the instances to be explained (left) and number of these instances per class (right) of the running example.

4.1.2 Evaluation

Fidelity will be calculated on a separate explanation test set consisting of 10.000 instances where each feature has been sampled from a normal distribution with a standard deviation of 100. A scatter plot and the difference classifier class counts are shown in Figure 4.4.

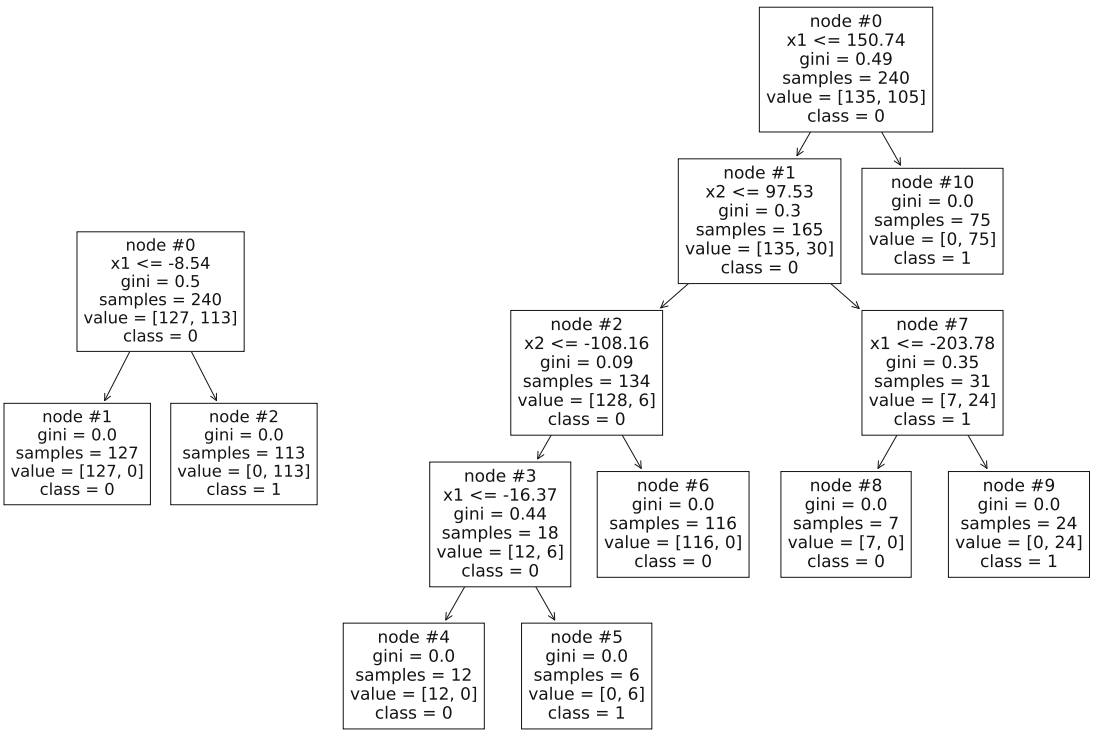


Figure 4.3: Classifier A's (left) and B's (right) decision nodes of the running example task.

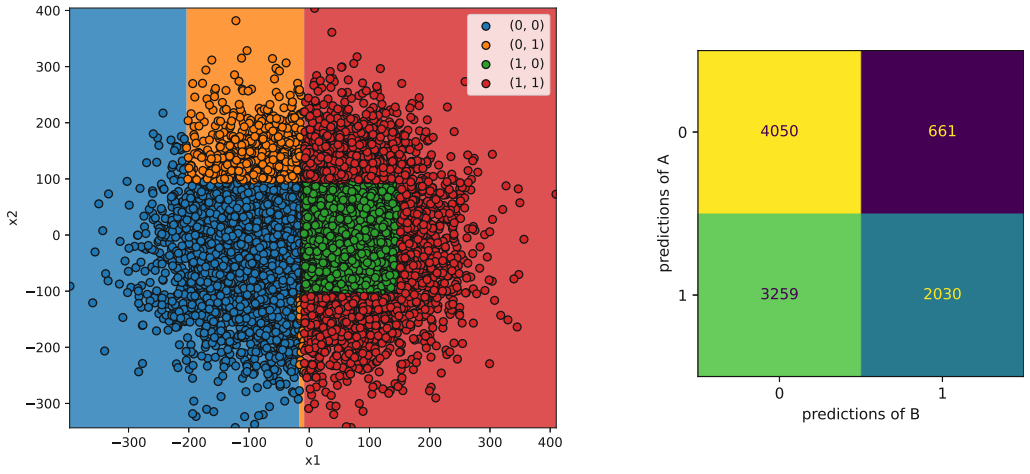


Figure 4.4: Difference classifier decision boundaries, overlaid by a scatter plot of the explanation test instances (left) and number of instances per class (right) of the running example.

4.2 One Classifier Ignores a Feature

To demonstrate the methods' workings in a slightly more complex setting, I developed a task based on a small, two dimensional data set and a binary target variable. Data set and the code for generating it can be found online[1]. The feature values of the instances were sampled from normal distributions and transformed differently for each label using randomly chosen parameters, such that both distributions are overlapping and no perfect separation is possible. The classifiers are of type logistic regression and thus inherently interpretable by their model weights. Their decision boundaries can be seen in Figure 4.5. Please note, that classifier B does not discriminate at $x_2 = 0$, but at approx. 22.1. Of the 300 instances, 150 have been used for training and 150 for evaluating classifier performance. The test set also serves for explanation generation.

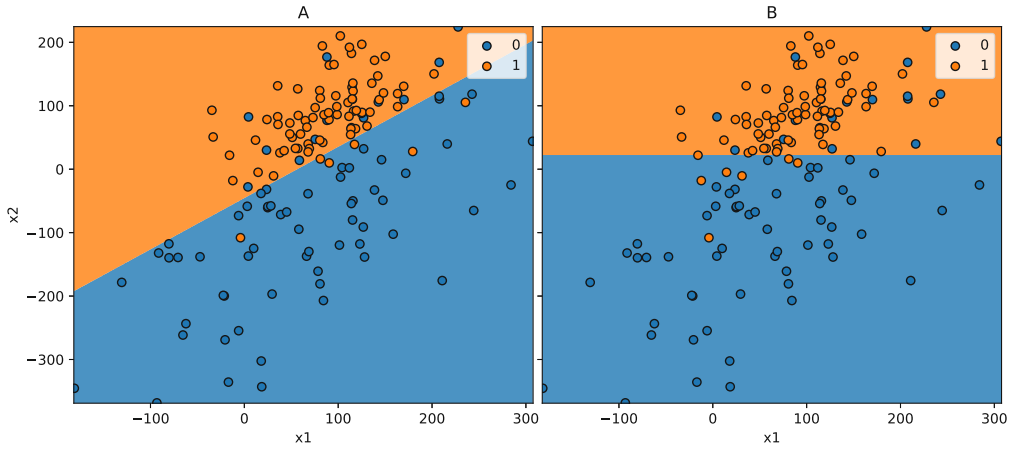


Figure 4.5: Each classifier's decision boundaries, overlaid by a scatter plot of the instances to be explained of the "One Classifier Ignores a Feature" example. The color denotes the true label as present in the original data set. Please note, that although the original data set is plotted for classifier B, it has only seen informative x_2 values during training.

In contrast to classifier A, B has been trained a modified version of the data set, where feature x_1 has been set to 0, thus making it non-informative to this classifier. The result is, that it can only rely on feature x_2 for decisions. As can be seen in the scatter plot including the decision boundaries of the difference classifier in Figure 4.6, there are two areas classified differently. In total there are 22 instances classified $(0, 1)$ and 15 instances classified $(1, 0)$. The classifiers offer probability estimates, that resemble their certainty about the predicted labels. In Figure 4.7 you can see a plot of the probability estimates of the individual classifiers and in Figure 4.8 a plot of the difference classifier's probability estimates.

[1]Data set available here: `https://doi.org/10.5281/zenodo.6502643`

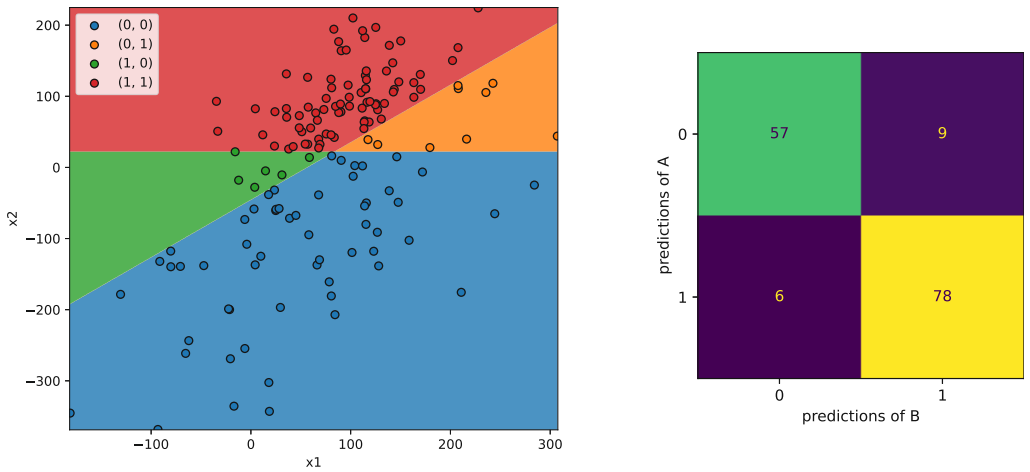


Figure 4.6: Difference classifier decision boundaries, overlaid by a scatter plot of the instances to be explained (left) and number of instances per class (right) of the "One Classifier Ignores a Feature" example.

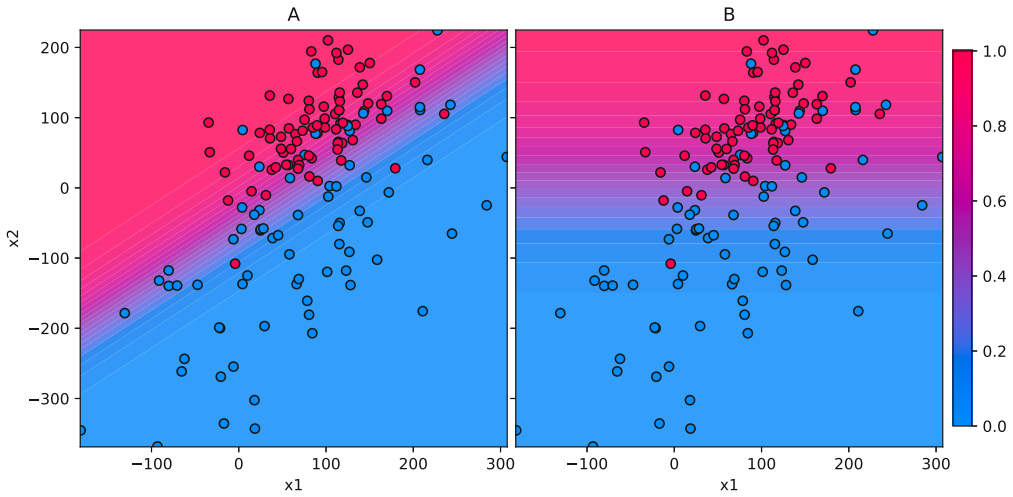


Figure 4.7: Probability estimates for class 1 of classifiers A and B of the "One Classifier Ignores a Feature" example. Instances are colored by their true label.

4.2.1 Ground Truth

The differences to be found can be defined with decision rules by using the regression weights of the classifiers. They can be seen in Table 4.1. Note, that β_1 of B is 0 and thus x_1 has no influence on the outcome. The rules can be derived by evaluating the regression model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ at $y = 0$ for each classifier and combining them afterwards. A log of odds value of zero means, that a classifier is predicting 50%, which is the exact decision boundary between the two labels of a binary classifier. For classifier B, I calculate the decision boundary with $x_2^* = -\beta_0/\beta_2 = 22.1$. For classifier A, I calculate the

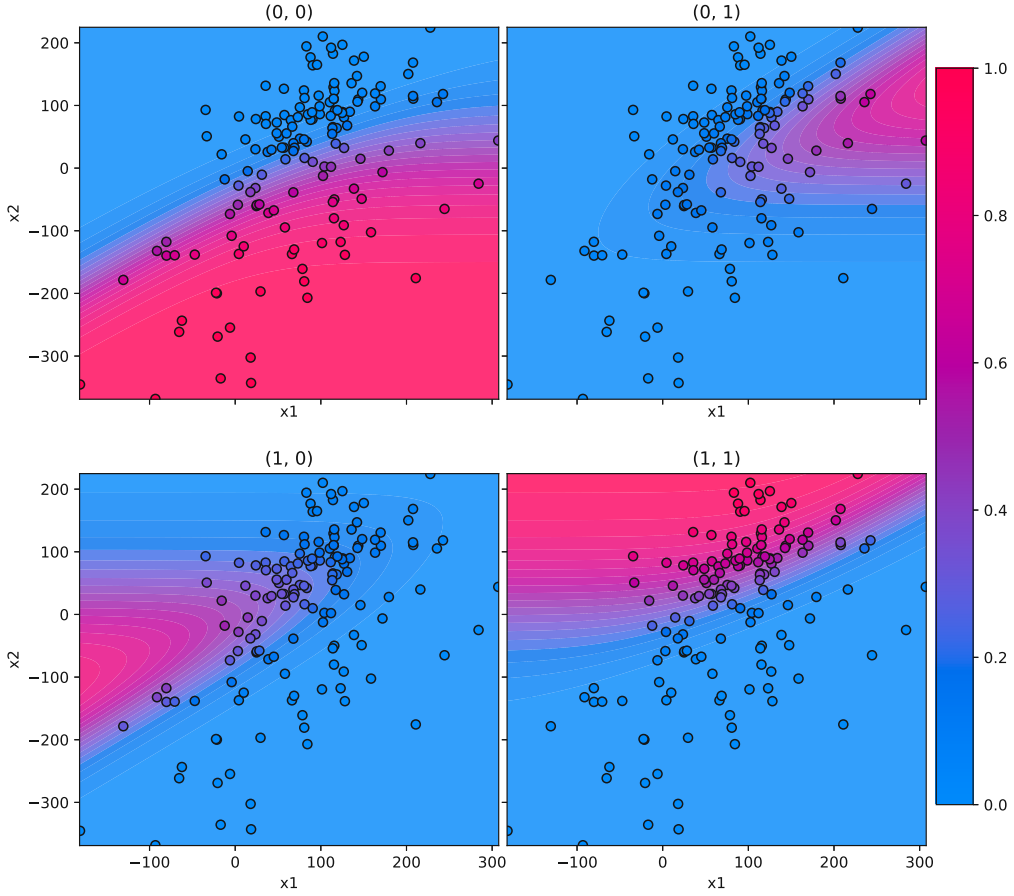


Figure 4.8: Probability estimates for all classes of the difference classifier of the "One Classifier Ignores a Feature" example. Instances are colored by their estimated probability.

decision boundary of x_2 as a function of x_1 with $x_2'(x_1) = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2}x_1 = -45.4 + 0.808x_1$. Combining this knowledge, I get the decision rules listed below. The feature thresholds have been rounded to one decimal and factors to three decimals for display.

	β_0	β_1	β_2
A	1.317	-0.023	0.029
B	-0.379	0	0.017

Table 4.1: Learned weights of the logistic regression classifiers.

1. if $x1 > 83.7 \wedge 22.1 < x2 \leq -45.5 + 0.808x1$, then label $(0, 1)$
2. if $x1 \leq 83.7 \wedge -45.5 + 0.808x1 < x2 \leq 22.1$, then label $(1, 0)$
3. else, then the classifiers predict the same labels

4.2.2 Evaluation

Fidelity will be calculated on a separate explanation test set consisting of 10.000 instances where each feature has been sampled from a normal distribution with a standard deviation of 100. They were centered to the crossing point between classifier A and B's decision boundaries. See the confusion matrix and scatter plot in Figure 4.9.

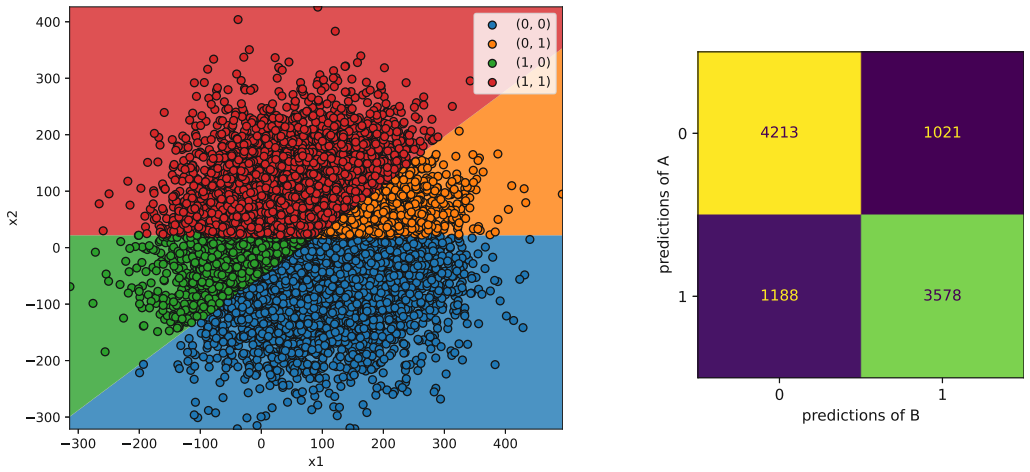


Figure 4.9: Difference classifier decision boundaries, overlaid by a scatter plot of the explanation test set instances (left) and number of instances per class (right) of the "One Classifier Ignores a Feature" example.

4.3 Gaussian Quantiles

This model comparison task and the data set have been originally published by Staufer and Rauber [SR21a]. I have set another parameter of the classifiers, that enables probability predictions and I changed the train/test split to increase the number of instances in the explanation data set. It is an artificially generated data set with two features. The instances were sampled from a two-dimensional normal distribution and separated by a concentric circle, so that approx. half of the instances are assigned label 0 and the other half label 1. It is split randomly into a training and a test set by sampling 50% each. A Support Vector Classifier (SVC) with a radial basis function kernel is trained and evaluated on this data, subsequently called classifier A. It will be compared to classifier B, another SVC, which is trained on a different data set, that has been generated with a different covariance parameter. But for generating explanations, the test set of the first data set is used. See Figure 4.10 for the classifiers' decision boundaries. The result is, that 55 instances are classified $(1, 0)$ which are aligned in the shape of a ring, as seen from a scatter plot of x_1 vs. x_2. See the decision boundaries of the difference classifier and instance counts per class in Figure 4.11. The plots for the probability estimates of the difference classifier are shown in Figure 4.12.

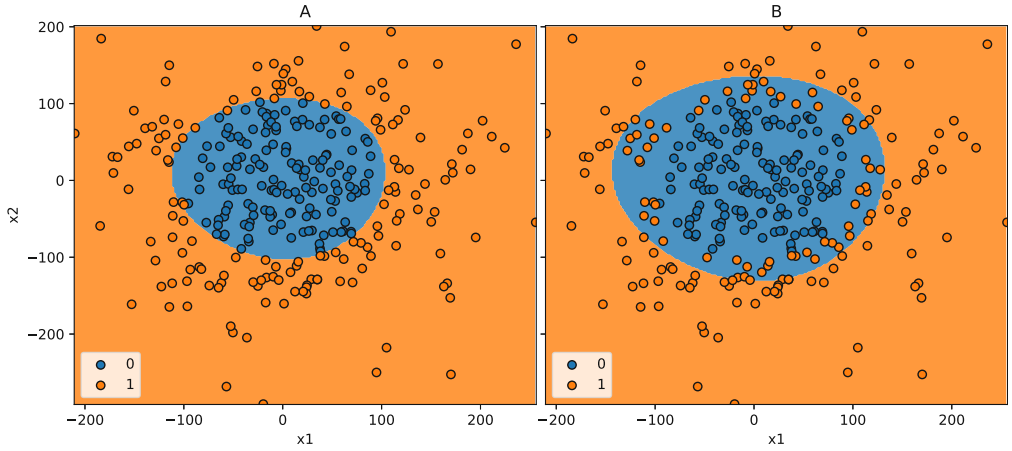


Figure 4.10: Each classifiers' decision boundaries, overlaid by a scatter plot of the instances to be explained of the Gaussian Quantiles example. The color denotes the true label as present in the original data set.

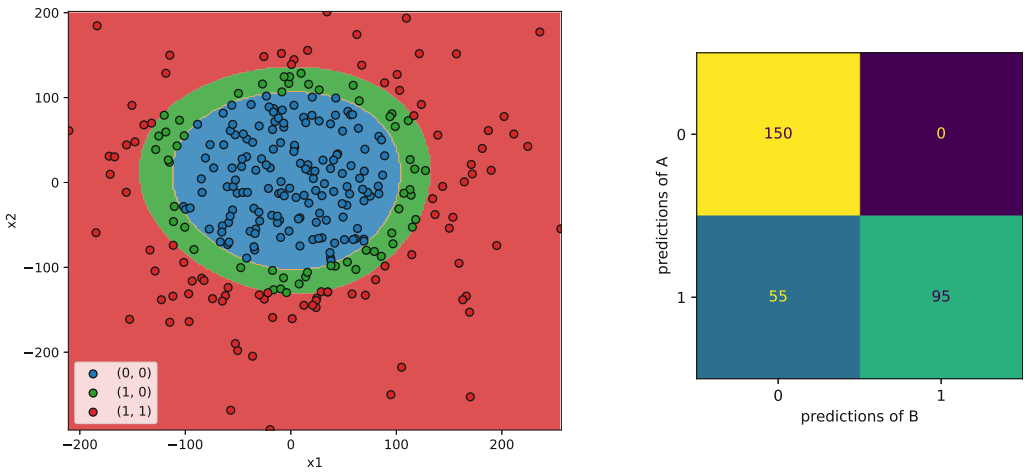


Figure 4.11: Difference classifier decision boundaries, overlaid by a scatter plot of the instances to be explained (left) and number of instances per class (right) of the Gaussian Quantiles example.

4.3.1 Ground Truth

Because the individual classifiers are too complex to be interpretable by model internals, I cannot define specific rules that resemble a ground truth. As can be seen from the plots in Figure 4.10, the classifiers didn't learn an exact circular shape, but a distorted version of it. So I need to rely on the knowledge, that the differences are arranged in the shape similar to a ring.

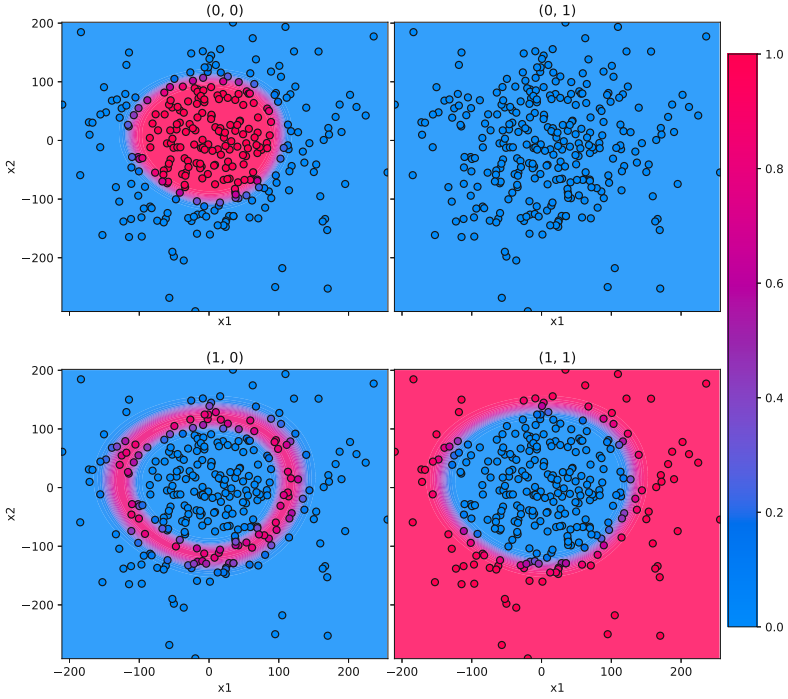


Figure 4.12: Probability estimates for all classes of the difference classifier of the Gaussian Quantiles example. Instances are colored by their estimated probability.

4.3.2 Evaluation

Fidelity will be calculated on a separate explanation test set consisting of 10.000 instances where each feature has been sampled from a normal distribution with a standard deviation of 100. A scatter plot and the confusion matrix are shown in Figure 4.13.

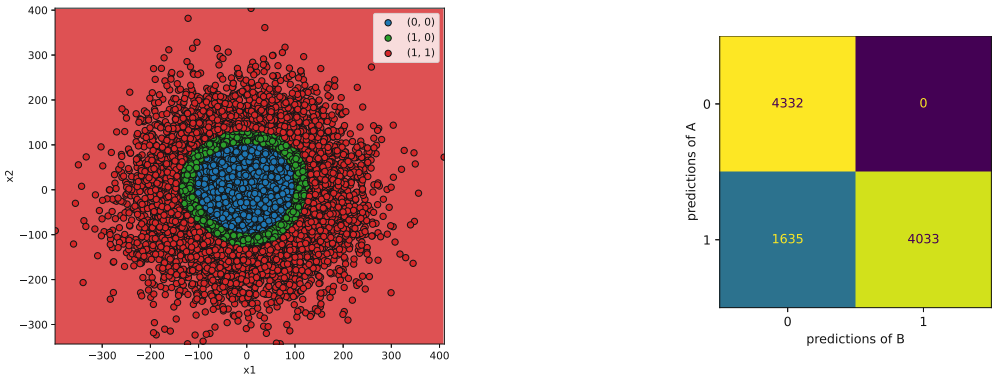


Figure 4.13: Difference classifier decision boundaries, overlaid by a scatter plot of the explanation test set instances (left) and number of instances per class (right) of the Gaussian Quantiles example.

4.4 Census Income (Adult)

This is the first real-world data set, that is used to demonstrate the interpretability methods in a benchmark setting. It can be found in the UCI machine learning repository [DG17]. The original goal for this data set is to predict whether an individual's income exceeds \$50,000 a year. There are twelve features of which eight are categorical and four are continuous. The target is binary and has the labels $False$ ($\leq$\$50,000 income) and $True$ (>\$50,000 income). The data set consists of 32,561 instances of which 70% are used for training the classifiers, 15% for evaluating their performance, 100 instances for generating explanations and 4785 instances for evaluating the explanation performance. The classifiers used are XGBoost [CG16] classifiers, which are ensembles methods based on decision trees. Generally, ensemble methods have a high complexity and their internal structure is not considered interpretable [MCB20]. The classifiers natively support probability estimates per class. To introduce differences, I will identify an influential feature, that can be modified and is then expected to show up in the explanations.

In Figure 4.14 you can see the feature importances, as calculated from model internals of classifier A. The calculation is based on the relative contribution a feature has on the resulting trees, called gain. This procedure is equal to that for Random Forests [AR19]. I choose *Hours per week* as the feature to be modified, because it is in the mid-range and therefore does not introduce too many differences when modified and is a continuous feature, which can be easily modified by adding or subtracting constant values. Adding 10 (hours) and training classifier B on the modified version results in five instances classified $(True, False)$ in the data set to be explained. In the explanation test set, this results in 238 instances classified $(True, False)$ and 8 classified $(False, True)$. So there is a clear tendency for B to predict label $False$ more often than A. See the counts per class in Figure 4.15.

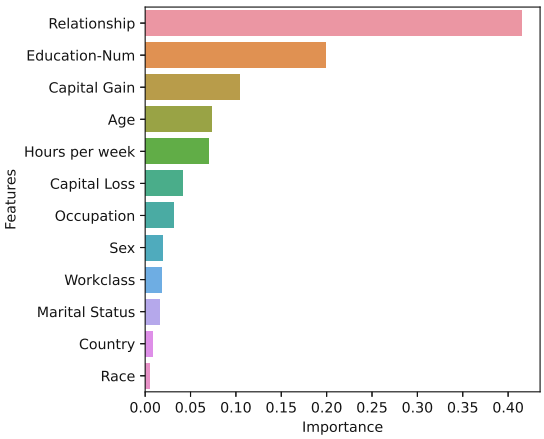


Figure 4.14: Feature importance of Classifier A, trained on the original Adult data.

The feature importances did not change in the retrained classifier with the modified

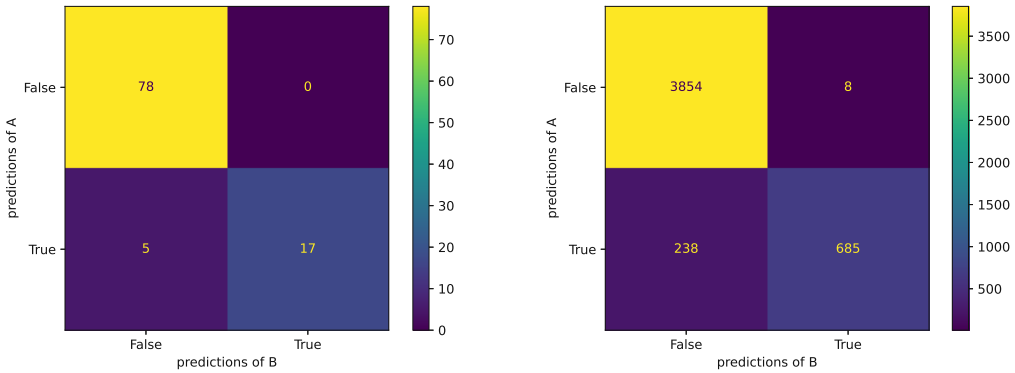


Figure 4.15: Difference classifier class counts of the data set to be explained (left) and explanation test set (right) of the Adult example.

index Feature	34	49	53	54	60
Age	50	32	46	32	42
Workclass	4	4	4	4	4
Education-Num	13	13	10	13	10
Marital Status	2	2	2	2	4
Occupation	5	12	14	12	4
Relationship	4	4	4	4	0
Race	4	4	4	4	4
Sex	1	1	1	1	0
Capital Gain	0	0	0	0	0
Capital Loss	0	0	0	0	2444
Hours per week	45	40	48	44	40
Country	39	39	39	39	39

Table 4.2: Instances to be explained, classified $(True, False)$, of the Adult example.

training data set. This is because the data transformation is of a linear nature. Classifier B achieves a very similar accuracy of 86% compared to classifier A with 87% on A's test set. In a practical scenario, 1% accuracy difference might not be enough of a reason to choose one classifier. You can see the instances to be explained, that are classified $(True, False)$, in Table 4.2. They have different values in features *Age*, *Education-Num* and *Hours per week*. But all of them have the same feature value for *Workclass*, *Race*, *Capital Gain* and *Country*. Instance #60 is furthermore different in features *Marital Status*, *Relationship*, *Sex* and *Capital Loss*.

Because of the complexity of the classifiers, it is not possible anymore to know all the effects that this modification has. But I expect the explanations to include the feature *Hours per week* as one reason amongst possible others.

4.5 Boston Housing

This is the second real-world data set. It was first published by Harrison and Rubinfeld [HJR78]. The original version can be downloaded from StatLib[2]. The original target is to predict house prices and thus a continuous variable, but it has been divided up into the three classes 0, 1 and 2. It is the first multiclass classification task used in the experiments. The two features *B* and *CHAS* are removed due to an ethical problem. There is one ordinal feature, *AGE*, which has three levels. The other features are continuous numbers.

In this experiment, the three classifiers A, B and C are compared. Thus it is also the first experiment, in which more than two classifiers are compared. They are all of type Multilayer Perceptron (MLP), a kind of neural network. This type is generally considered as not being interpretable [Mol20]. The implementation shipped with scikit-learn [PVG+11] is used. Each classifier has the same configuration, including 16 neurons and no hidden layers, which means they are only capable of representing linear separable functions. Each is further configured with a logistic sigmoid activation function and a regularization term set to 10^{-5}. The classifiers natively support probability and log-transformed probability estimates per class.

Classifier A is trained on the unmodified version of the data set, whereas B and C are trained on modified data. In B's training data, *AGE* is inverted: a value of 0 turns to 2, a value of 2 turns to 0 and a value of 1 stays the same. In C's training data, the true labels of a group of instances are changed. These instances are part of a cluster of a clustering based on the three features *LSTAT*, *ZN* and *CRIM*. The instance numbers per difference class are shown in Figure 4.16 for the two comparison pairs A vs. B and A vs. C. In Tables 4.3 and 4.4 the feature values are shown for all instances classified differently. I expect from the interpretability methods that they include the features used in the manipulations in the explanations. In the comparison of A and B I further expect global-level understanding of the differences and in the comparison of A and C I expect them to narrow the differences down to a certain area in the feature space. There is no test data set available in this case, but I evaluate with regards to the knowledge about the modifications.

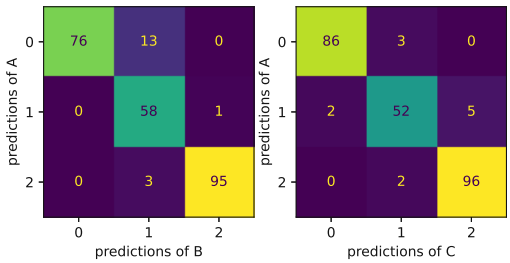


Figure 4.16: Difference class counts of classifiers A vs. B (left) and A vs. C (right) for the explanation data set, Boston Housing example.

[2] http://lib.stat.cmu.edu/datasets/boston, accessed on 12 March 2022

index	Label	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
8	(0, 1)	3.8	0	18.10	0.77	6.25	2	2.30	24.0	666	20.2	14.19
28	(0, 1)	2.2	0	19.58	0.60	5.85	2	2.42	5.0	403	14.7	11.64
40	(0, 1)	2.3	0	19.58	0.60	6.32	2	2.10	5.0	403	14.7	11.10
42	(0, 1)	1.1	0	8.14	0.54	5.70	2	3.82	4.0	307	21.0	18.26
74	(0, 1)	0.6	0	21.89	0.62	5.73	2	2.07	4.0	437	21.2	17.25
151	(0, 1)	0.9	0	8.14	0.54	6.02	2	4.44	4.0	307	21.0	17.07
152	(0, 1)	2.3	0	19.58	0.60	5.88	2	2.39	5.0	403	14.7	12.03
157	(0, 1)	1.0	0	21.89	0.62	5.76	2	2.35	4.0	437	21.2	17.31
180	(0, 1)	0.3	0	21.89	0.62	5.69	2	1.79	4.0	437	21.2	17.19
196	(0, 1)	0.9	0	8.14	0.54	5.61	2	4.35	4.0	307	21.0	16.80
238	(0, 1)	0.0	0	13.89	0.55	5.89	1	3.11	5.0	276	16.4	13.51
244	(0, 1)	0.3	0	21.89	0.62	5.69	2	1.81	4.0	437	21.2	17.35
245	(0, 1)	1.2	0	8.14	0.54	6.14	2	3.98	4.0	307	21.0	18.72
51	(1, 2)	0.3	0	7.38	0.49	6.31	0	5.42	5.0	287	19.6	6.15
34	(2, 1)	4.6	0	18.10	0.72	3.56	2	1.61	24.0	666	20.2	7.12
77	(2, 1)	0.1	33	2.18	0.47	6.62	1	3.37	7.0	222	18.4	8.93
101	(2, 1)	0.1	0	11.93	0.57	6.79	2	2.39	1.0	273	21.0	6.48

Table 4.3: Instances to be explained, classified differently by A and B, of the Boston Housing example.

index	Label	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
8	(0, 1)	3.8	0	18.10	0.77	6.25	2	2.30	24.0	666	20.2	14.19
40	(0, 1)	2.3	0	19.58	0.60	6.32	2	2.10	5.0	403	14.7	11.10
238	(0, 1)	0.0	0	13.89	0.55	5.89	1	3.11	5.0	276	16.4	13.51
23	(1, 0)	0.8	0	8.14	0.54	5.60	2	4.45	4.0	307	21.0	16.51
156	(1, 0)	0.8	0	8.14	0.54	5.65	2	4.45	4.0	307	21.0	16.48
22	(1, 2)	0.1	60	1.69	0.41	6.58	0	10.71	4.0	411	18.3	5.49
55	(1, 2)	0.5	20	3.97	0.65	7.23	1	2.12	5.0	264	13.0	9.52
68	(1, 2)	0.0	90	2.97	0.40	7.09	0	7.31	1.0	285	15.3	7.85
122	(1, 2)	0.0	85	4.15	0.43	6.52	0	8.54	4.0	351	17.9	6.36
206	(1, 2)	0.1	80	1.91	0.41	5.94	0	10.59	4.0	334	22.0	5.57
34	(2, 1)	4.6	0	18.10	0.72	3.56	2	1.61	24.0	666	20.2	7.12
101	(2, 1)	0.1	0	11.93	0.57	6.79	2	2.39	1.0	273	21.0	6.48

Table 4.4: Instances to be explained, classified differently by A and C, of the Boston Housing example.

CHAPTER 5

Mocca-SHAP

In this work, I propose the classifier comparison method Model comparison with clustered difference classifier SHAP values (Mocca-SHAP). At its core is the instance-level interpretability method SHAP [LL17], which enables global and modular level explanations with SHAP Dependence Plots [LEL19]. Two types of explanations can be interpreted: traditional, side-by-side explanations for the target classifiers and explanations for the difference classifier. The difference classifier is an intermediate model that merges the outputs of two classifiers that are to be compared. It has been proposed by Staufer and Rauber [Sta21] and I am extending it to support SHAP. It further offers an interactive way to investigate modular parts of the explanation space in a hierarchical way, based on the *supervised clustering* approach of Lundberg [LEL19]. To enable causal interpretation, group counterfactual explanations are included in each modular explanation.

In this chapter, I describe Mocca-SHAP's requirements and limitations, the design process about how I incrementally develop and integrate findings from the experiments. Then I show how quantitative quality metrics can be automatically derived from explanations of Mocca-SHAP, DiRo2C and a baseline. Finally, I describe details about the implemented tools and conducted experiments, and where to find them.

5.1 Requirements and Limitations

Mocca-SHAP focuses on classifiers, so it does not support comparing regression models or non-predictive models. But it supports any kind of classifier, by treating them as black-boxes. It supports comparing binary as well as multiclass classifiers. Multioutput classifiers are out of scope for this work. I also restrict the data types to tabular data sets in this work, although SHAP could support image or other types as well. The tools I implement take as input an explanation data set and require access to the prediction function of two trained classifiers with a *scikit-learn*-like interface [PVG+11]. These classifiers are subsequently referred to as A and B. The true labels of the data set are

not required. The output space can be chosen based on the capabilities of the compared classifiers. The algorithm used to compute SHAP Values in the experiments is the optimized exact algorithm, but can be changed for another. Only two classifiers can be compared at a time. To compare more than two, the task needs to be broken down into multiple pairwise comparisons, as is done in the Boston Housing experiment.

5.2 Design Process

5.2.1 Probabilistic Extension of the Difference Classifier

Some interpretability methods like SHAP are able to leverage probability estimates of classifiers. Staufer and Rauber [Sta21] have only proposed the binary and multiclass difference classifier variants that predict labels based on simple rules, which I extend for class probability estimates. As mentioned in Section 2.1, probabilistic classifiers have to meet two requirements: (1) the label with the highest estimated class probability has to match the predicted label and (2) that the class probability vector sums up to 1.

By treating the probability estimates of the classifiers as independent events, I can calculate the joint certainty of a specific class of the multiclass difference classifier easily by multiplying two probability estimates, as shown in Equation 5.1. i, j are classes of the individual classifiers, (i, j) is a class of the multiclass difference classifier and $\hat{y}_d^{(i,j)}$ the merged probability of the difference classifier for a specific class. Both requirements for classifiers are satisfied: (1) summing up the merged probabilities for all class combinations always results in 1, and (2) the class label with the highest probability matches the predicted label as determined by the rules shown in the last section.

$$\hat{y}_d^{(i,j)} = \hat{y}_A^i \hat{y}_B^j \tag{5.1}$$

Yet I have not found a solution for the binary difference classifier. It is not sufficient to build on the probability estimates of the multiclass difference classifier and sum up all difference class estimates into one number and all equality class estimates into one number, because then there is a mismatch between the predicted label as determined by the rules shown in Section 2.6.1 and the highest estimated class probability. Consider e.g. a classification problem with the three classes 0, 1 and 2: Classifier A predicts for an instance the class probability vector $(0.5, 0.3, 0.2)$ and classifier B $(0.5, 0.2, 0.3)$. Thus, both A and B predict the label 0, because it has the highest class probability. The multiclass difference classifier has the classes $(0, 0)$, $(0, 1)$, $(0, 2)$, $(1, 0)$, $(1, 1)$, $(1, 2)$, $(2, 0)$, $(2, 1)$ and $(2, 2)$, and estimates a class probability vector of $(0.25, 0.15, 0.1, 0.15, 0.06, 0.09, 0.1, 0.04, 0.06)$ for this instance. Because the predicted label is 0 (equal), the estimated probability for this class must also be higher than that for the other class. But actually, the sum of the probability estimates for the equality classes is lower than that for the difference classes ($0.37 < 0.63$). This violates one of the requirements for probabilistic classifiers and therefore the binary difference classifier is not suited for use with the chosen interpretability method SHAP.

Other Variations of the Multiclass Difference Classifier

Other variations would be possible as well, like simplifying the multiclass difference classifier by combining only the equality classes into a single class and leaving the difference classes separate. But then again the requirements for probabilistic classifiers would not be satisfied anymore.

Difference Classifier with Truthfulness

Another variation could take the true labels into account, with the additional assumption that they are known for the explanation data set. Now $\hat{y}$ is a predicted label of a classifier for an instance with the true label y. See below the rules for determining the label in the simplest case with binary classifiers:

- if $y = 0 \wedge \hat{y}_A = 0 \wedge \hat{y}_B = 0$, then label 0
- if $y = 0 \wedge \hat{y}_A = 0 \wedge \hat{y}_B = 1$, then label 1
- if $y = 0 \wedge \hat{y}_A = 1 \wedge \hat{y}_B = 0$, then label 2
- if $y = 0 \wedge \hat{y}_A = 1 \wedge \hat{y}_B = 1$, then label 3
- if $y = 1 \wedge \hat{y}_A = 0 \wedge \hat{y}_B = 0$, then label 4
- if $y = 1 \wedge \hat{y}_A = 0 \wedge \hat{y}_B = 1$, then label 5
- if $y = 1 \wedge \hat{y}_A = 1 \wedge \hat{y}_B = 0$, then label 6
- if $y = 1 \wedge \hat{y}_A = 1 \wedge \hat{y}_B = 1$, then label 7

But this restricts the types of interpretability methods that can be used to explain the task. Methods like SHAP and DiRo2C are based on perturbing instances and measuring the effect on the outcome. Thus, they need to be able to test arbitrary input to the classifier, but this problem formulation allows to only predict instances with known true labels. There is a workaround: By actually training an intermediate classifier and not relying on conversion rules the task could be approximated. But as this introduces an additional error, I do not investigate it further in this work.

5.2.2 Selection of a Global Extension Method

SHAP Dependence Plots have been noted in Section 2.4.5 to be one of the global extension methods, based on SHAP Values. For a couple of reasons, I selected them as the main global interpretability method for my work. (1) They allow for interpreting the relation between an input feature and its effect on the outcome, assuming that all other features are fixed. This is known as the marginal effect of a feature, and is widely used with PD-plots and ALE-plots. (2) Compared to PD-plots, SHAP Dependence Plots additionally

visualize interaction effects via vertical spread in the scatter plot. By additionally coloring each instance by another feature value allows to interpret, to what extent interaction effects with that feature account for deviations in the effect. (3) Compared to SHAP Summary Plots, the feature-output relation is easier to interpret. While these plots actually contain the same information, it is in different dimensions. [LEL19]

5.2.3 Explanations for the Individual Classifiers

The traditional way of comparing classifiers with SHAP is to generate SHAP Values individually for each classifier, create plots side-by-side and interpret what is different. For binary classifiers, it is enough to visualize one of the two classes. For multiclass classifiers, each class needs a separate plot.

Consider the dependence plots for classifier A and B of x_1 of the running example (see Section 4.1 for the task description), shown in the left two scatter plots in Figure 5.1. Ignore the coloring for now. You can see the actual feature values of x_1 on the x-axis and the SHAP Values of x_1 ($s(x_1)$) on the y-axis. The SHAP Values are in the units of the model output space they were created for, which is probability in this case. Each dot corresponds to one instance of the data set. This allows me to interpret, that x_1 has a step-like relation with its effects: Up to a certain point, it has a constant decreasing effect on the outcome of approx. -0.5 (or -50%), while above approx. $x_1 > 0$ it has an increasing effect of approx. 0.5 (or +50%). The effect of x_1 for classifier B is different. There is a more gradual change from a decreasing effect of low feature values to an increasing effect for high feature values. Also, for some instances it has a different effect while having the same feature value. This can be seen as vertical spread. It is the result of interaction with other features - in this case x_2, as it is the only other feature. x_2 has an influence on x_1's effect. In contrast, there is no vertical spread visible in classifier A's dependence plot.

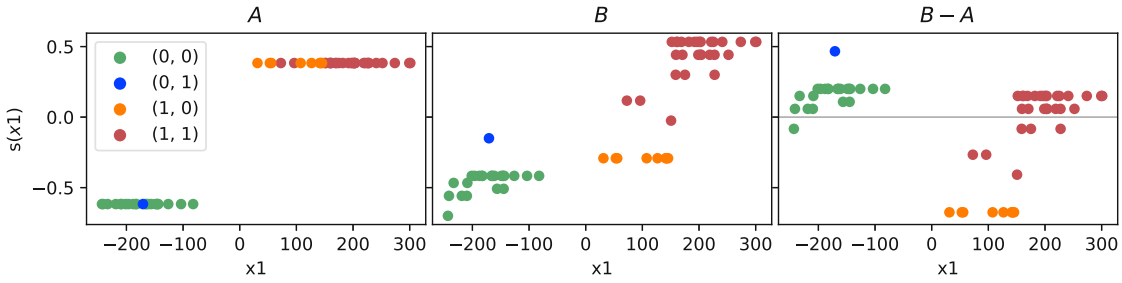


Figure 5.1: Individual classifiers' SHAP Dependence Plots (left and middle) and difference dependence plots (right) for x_1's effect on class 1's probability outcome, running example.

Coloring instances by their predicted class: As Miller [Mil19] notes, *"probabilities are not as important as causal links"*. Adding another dimension by coloring the instances depending on their difference classifier label, causal links can be established between probabilities and difference classes. Have a look again at the two left plots in Figure 5.1.

I have coloured each instance by its actual class. Each class name is made up of A's and B's actually predicted label.

E.g. class $(0, 1)$ (blue) means, that A predicts class 0 and B class 1. There is only one instance coloured blue, and it deviates from other instances with similar x_1 values (lower than 0), in that it has a less decreasing effect on classifier B's outcome than the others. The effect on A is not different from similar instances, which are of class $(0, 0)$. It allows me to reason, that x_1's attribution is not decreasing the probability outcome as much in a certain case as for other instances with $x_1 < 0$. Considering, that the vertical deviation is the result of another feature influencing x_1's effect, it has to be a combination of another feature and x_1's effects that causes B to classify this particular instance differently than A.

Now consider class $(1, 0)$ (orange). On B, these instances are in a straight horizontal line, each having a decreasing effect, while the other instances with $x_1 > 0$ have either an effect close to 0 or an increasing effect. Just like for the instance classified $(0, 1)$, the other feature influences x_1's attribution.

Difference dependence plots: To ease interpretation, I added a third plot to A and B's dependence plots, which shows the difference between A and B's SHAP Values. You can see it in the right plot in Figure 5.1. It allows to interpret, to what extent and for which feature values classifier B overestimates (positive differences) and underestimates (negative differences) classifier A. Overestimation in probability space means, that the effect a feature has is increasing the predicted probability for the class, thus forcing the prediction more towards this class. Underestimation means, that it is forcing the prediction away from this class.

A grey horizontal line at zero makes it easier to distinguish over- and underestimation. For multiclass classification tasks like in the Boston Housing experiment, I switched to placing the plots per feature on top of each other in order to be able to visualize the different classes' effects in columns. This behaviour is implemented in the prototypical tools, because it supports both types in a general way.

Now have a look again at the right plot in Figure 5.1. You see, that the instances below a feature value of 0 all have a SHAP Value difference close to 0, except the instance in blue. It is actually classified 1 by B and 0 by A. Classifier B overestimates its effect, according to this plot, because it has a high positive SHAP Value difference. This can actually be considered a satisfactory explanation for humans, because it explains, why this abnormal classification of the blue instance might have happened in contrast to the normal classifications of the green instances. The picture is not yet complete, because we are looking at one feature only.

Consider the instances classified $(1, 0)$ in the difference dependence plot in Figure 5.1. They have abnormally low SHAP Value differences, thus, B is underestimating the effect on class 1. This explanation is satisfactory, because it explains why it decides against

class 1 and predicts class 0. If it would have been higher, like for the instances in red, it might have predicted class 1 just like classifier A.

5.2.4 Explanations for the Difference Classifier

Besides the individual explanations, Mocca-SHAP includes explanations for the difference classifier, which are not to be confused with the SHAP Value differences. The difference classification task includes m^2 classes, with m being the number of target classes of the original classification task. They relate to the confusion classes if comparing classifier A's to B's predicted classes. Therefore, dependence plots can be created for each class and feature. The number of plots poses a potential problem.

Problem break-down: One way to tackle this is by reducing the number of plots one has to interpret at a time. One class can be put into focus and compared to a selection of other classes. It is sufficient to repeat this for all difference classes, but equality classes can also be put into focus. It depends on the task, which other classes to select. Classes without instances may be skipped.

In Figure 5.2 you can see the difference classifier dependence plots with class $(0, 1)$ in focus (middle). I have chosen classes $(0, 0)$ and $(1, 1)$ for reference and omitted class $(1, 0)$. Ignore the counterfactual legend and vertical lines for now. Each dependence plot shows the effect, that feature x_1 has on the joint predicted probability, that a certain class combination has. The middle plot shows the effect on A's predicted probability for class 0 times B's predicted probability for class 1. In the same manner, the other two plots show joint effects for the classes $(0, 0)$ and $(1, 1)$.

- From the left plot, I interpret that values of x_1 smaller than 0 have an increasing effect on the probability that A predicts for class 0 and that B predicts for class 0 (or difference classifier class $(0, 0)$'s outcome for short). In contrast, higher values have a decreasing effect. One instance stands out, the instance in blue. For it, the increasing effect is lower.
- From the middle plot, I interpret that there is a constant effect close to zero for feature values all over the range. Except for the instance in blue, because for it x_1 has an increasing effect on the probability that A predicts for class 0 and B for class 1 (or difference classifier class $(0, 1)$'s outcome for short).
- From the right plot, I interpret that x_1 values lower than 0 have a decreasing effect on the probability that A predicts for class 1 and that B predicts class 1 (or difference classifier class $(1, 1)$'s outcome for short). For values between 0 and approx. 150, there are instances for which it has a decreasing effect and instances, for which it has an increasing effect. But for now it is only relevant that higher value than 150 have a clear increasing effect on the probability that A predicts for class 1 and B for class 1.

By combining the knowledge now gained about the instance classified 0 by A and 1 by B, I reason that this is because of another feature influencing the effect of x_1 to be more increasing on the class $(0, 1)$'s outcome and at the same time less increasing on the class $(0, 0)$'s outcome.

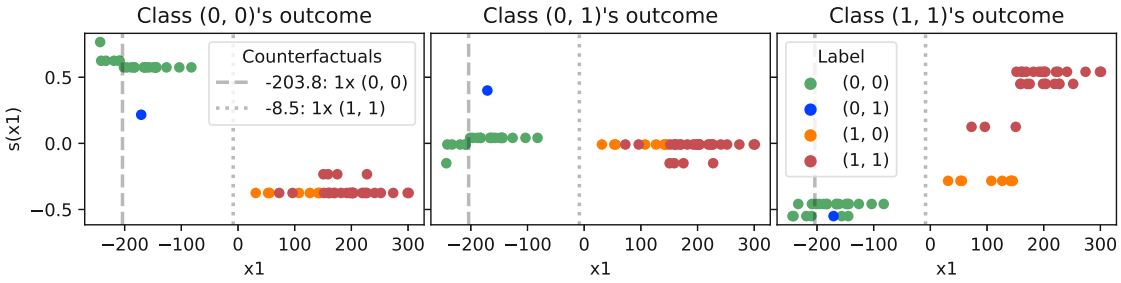


Figure 5.2: Difference classifier dependence plots for x_1 with class $(0, 1)$ in focus, running example.

Group counterfactual explanations: Still, I am not able to answer the most important question: "What needs to change for certain instances which are currently classified differently to be classified equally?", which refers to the need of humans for contrastive explanations [Mil19]. To answer this question, I propose *group counterfactual explanations*. Each consists of a single feature value, that, if replacing all feature values in the group of instances with it, flips each instance's prediction. Single feature counterfactual explanations are the preferred type according to Verma et al. [VDH20] because they are the simplest and easiest to understand. With the algorithm shown in 5.1, every feature is treated as a continuous variable and two group counterfactual explanations are searched for every feature: One denoting an upper boundary and one a lower boundary, which can be visualized in the dependence plots to support causal interpretation. The algorithm requires as input (1) the prediction function $f(x)$ of the difference classifier, (2) a group of instances X for which to find counterfactual explanations (3) a step size s at which to increase and decrease feature values, (4) the limits l^-, l^+ at which to stop the search, (5) the feature j for which to find the boundaries. It returns a lower and an upper feature value.

The computed counterfactuals are visualized as vertical lines in each dependence plot to mark the boundary. Have a look again at Figure 5.2, where the dashed line denotes the lower counterfactual for the instance in blue and the dotted line its upper counterfactual. From this I can reason, that within these two boundaries, the instance in blue is classified as class 0 by A and class 1 by B. Yet if its x_1 value would be -203.8, classifier B would flip and also predict class 0, just like A. If it was -8.5, classifier A would flip and also predict class 1, just like B.

Please note, that sometimes there is only one boundary, or none at all, depending on which counterfactuals the algorithm finds. Also, care needs to be taken when interpreting them, because it treats the feature as being a continuous variable. It can be applied to

categorical types as well, if they are encoded as numbers, but then it just searches for two arbitrary counterfactual explanations.

Algorithm 5.1: Computation of Group Counterfactual Explanations

Data: $f(x), X, s, l^-, l^+, j$
Result: b^-, b^+
$b^- \leftarrow min(X_j)$;
$b^+ \leftarrow max(X_j)$;
$y' \leftarrow f(X^{(0)})$;
while *any* $f(X)$ *is equal to* y' **do**
  $b^- \leftarrow b^- - s$;
  if $b^- \leq l^-$ **then**
    $b^- \leftarrow -\infty$;
    break
  end
  $X_j \leftarrow b^-$;
end
$X_j \leftarrow b^+$;
while *any* $f(X)$ *is equal to* y' **do**
  $b^+ \leftarrow b^+ + s$;
  if $b^+ \geq l^+$ **then**
    $b^+ \leftarrow \infty$;
    break
  end
  $X_j \leftarrow b^+$;
end

5.2.5 Local Explanations

Until now, I have put together all instance-level explanations (SHAP Values) to explain the global level. Global-level explanations are not suited to explain complex behaviour, but explanations with local validity are. Because all the extension methods of SHAP are just based on multiple instances' explanations, they can explain arbitrary subsets of explanations.

Consider again the running example, with the global dependence plots of the difference classifier shown in Figure 5.2. We know, that an interacting feature is causing the higher increase in effect of the probability of class $(0, 1)$ for low x_1 feature values, specifically for the instances in the range between the two counterfactuals. Because there is only one other feature, it is simple, which one to investigate further. I extract a subset of instances, based on the condition $-203.8 < x_1 < -8.5$. Now I create difference classifier dependence plots for this subset, now referred to as node α. I refer to it as a node, because it can be seen as part of a modular, hierarchical structure. The sibling node is called $\neg\alpha$. Plots

for the effect of x_2 for both nodes are shown in Figure 5.3. I include the sibling node just for reference, to get a more complete understanding of the data. We now can see, that the instance in blue is different from other instances in node α, in that it has a higher x_2 value. There is a step-like relation, with x_2 having a higher increasing effect on $(0, 1)$'s probability and bigger decreasing effect on $(0, 0)$'s probability. There are also other instances with higher x_2 values, as can be seen in the reference subset $\neg\alpha$. We can see one lower counterfactual at $x_2 = 97.5$, which helps me determine a boundary. Finally, I reason that instances fulfilling $-203.8 < x_1 < -8.5$ (part of node α) are classified 1 by B while being classified 0 by A, if and only if x_2 is greater than 97.5.

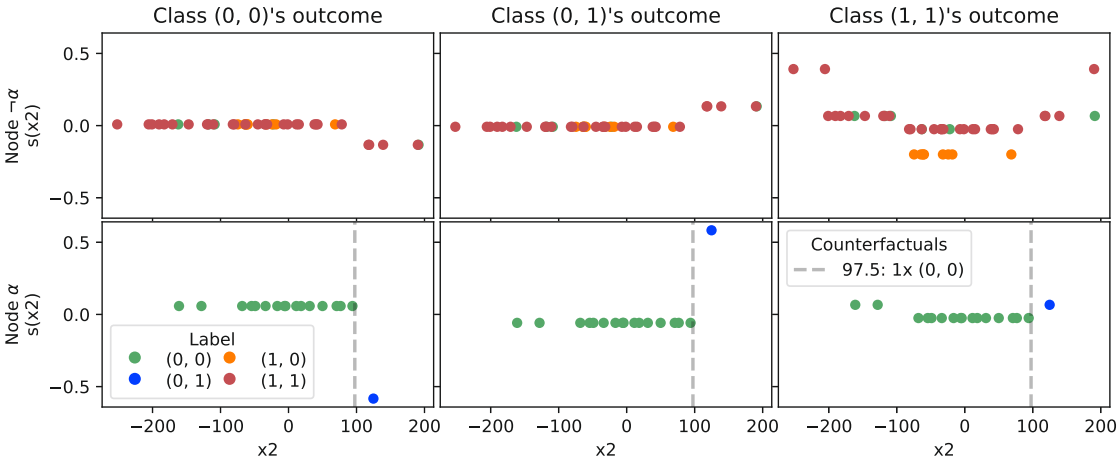


Figure 5.3: Difference classifier dependence plots for node α and all instances not part of node α with class $(0, 1)$ in focus, running example.

Supervised clustering (described in Section 2.4.5) groups instances by explanation similarity and because it exposes a hierarchical clustering structure, there is the possibility to investigate local explanations in a modular way. The tree-like structure can be descended in a top-down approach and allows for an interactive, conversational understanding of the explanations. As input to the clustering algorithm, I am passing the concatenated difference classifier SHAP Values.

With the interactive top-down approach, each lower (more detailed) explanation gets a context: that of the upper levels. Thus, I am addressing contextuality, the essential finding of Miller [Mil19]. He argues, that *"While an event may have many causes, often the explainee cares only about a small subset (relevant to the context), the explainer selects a subset of this subset (based on several different criteria), and explainer and explainee may interact and argue about this explanation."* [Mil19]. In this manner, I interpreted the explanations during the experiments.

Have a look at the difference class dependence plots for x_2 of the "One Classifier Ignores a Feature" example (see Section 4.2 for the task description) in Figure 5.4. It shows plots for class $(0, 1)$ in the middle, $(0, 0)$ on the left and $(1, 1)$ on the right, plus counterfactuals for the instances classified $(0, 1)$ (blue). The SHAP Values are now in log odds instead

of probabilities. I have picked three nodes from the clustering hierarchy and created dependence plots for them (rows 2-4), along with the plots for the global level (first row), termed "root node". The name of the nodes describes their path within the hierarchy. The letter 'L' stands for left child, 'R' for right child. Choosing the nodes is up to the data scientist. Here I first selected the nodes LR and LL, because they contained all instances classified $(0, 1)$. But it appeared to me, that node LR contained too diverse information, so I split it again into nodes LRL and LRR. Then, I descended each node a bit further to reduce the number of instances, but without loosing any of the instances classified $(0, 1)$, to finally arrive at the selection of nodes LLR, LRL and LRRRR.

On the global level (root node, first row) I interpret, that the effect of x_2 on the predicted probability of class $(0, 0)$ is increasing for low feature values and decreasing for high feature values, with the shape of the relation curve being monotonically decreasing. The effect on class $(1, 1)$ is the opposite. The effect on class $(0, 1)$ is shaped concave, with peak values having an increasing effect and the rest a decreasing effect. Now what does that explain about the instance classified $(0, 1)$? Because of the concave shape of the $(0, 1)$ curve, I can expect their increasing effect to lower if their x_2 value was higher or lower. If it was lower, the effect on $(0, 0)$ would increase. If it was higher, the effect on $(1, 1)$ would increase. The counterfactuals actually confirm this observation: at $x_2 = 22.1$, B now flips its prediction to conclude with A on class 0, while at $x_2 = 202.8$, A flips its prediction to conclude with B on class 1.

From the local dependence plots, I can see that the behaviour is different from the global view, because the upper counterfactual is different in each node. It is highest in node LRL and lowest in node LRRRR. But the lower counterfactual is the same for all. Now have a look at the scatter plots with decision boundaries in Figure 5.5, each depicting one of the local nodes. The decision boundaries are visualized by colouring the background in the color of the actually predicted class. The horizontal dashed lines show the x_2 counterfactuals of each node. You can see, that every node contains instances, that are close to one part of the decision boundary of $(0, 1)$, node LLR at the upper left boundary between $(0, 1)$ and $(1, 1)$, node LRL at the lower boundary between $(0, 1)$ and $(0, 0)$ and node LRRRR at the peak of $(0, 1)$ which is adjacent to $(0, 0)$, $(1, 1)$ and $(1, 0)$.

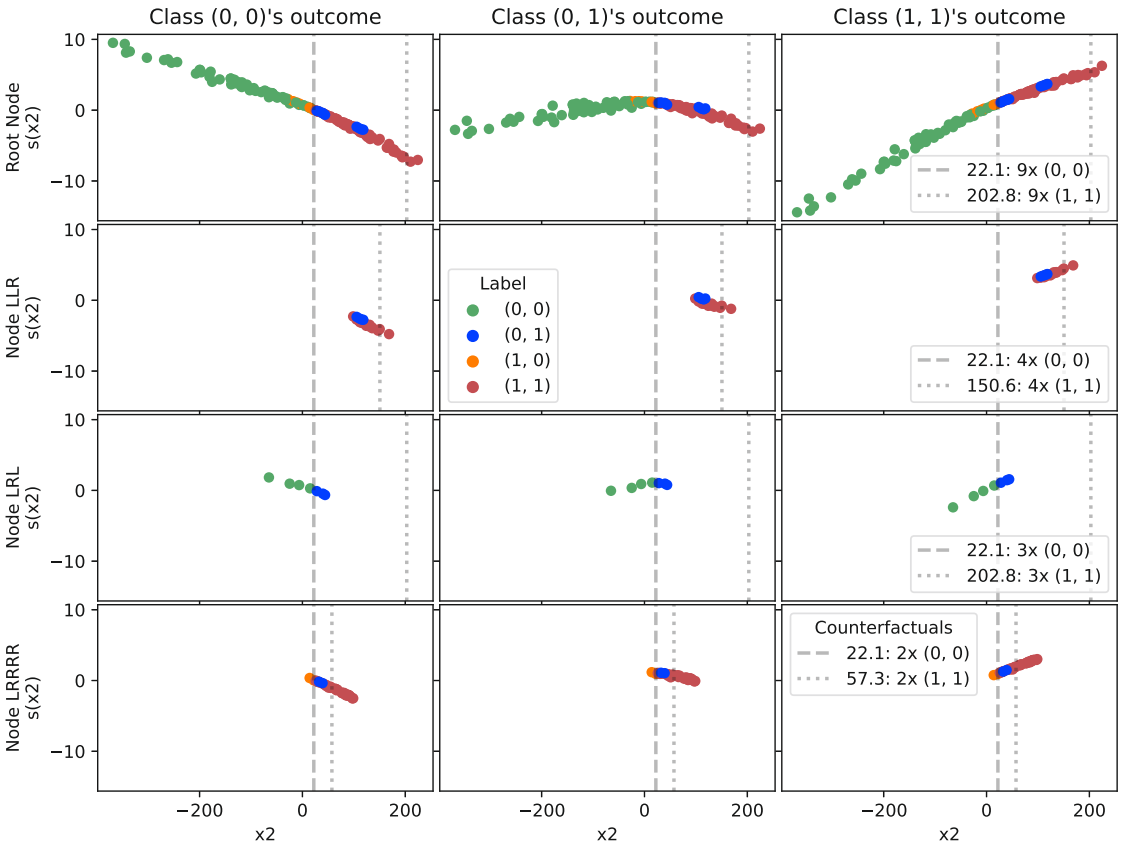


Figure 5.4: Difference classifier dependence plots of feature x_2 with instances classified $(0, 1)$ in focus (blue), "One Classifier Ignores a Feature" example.

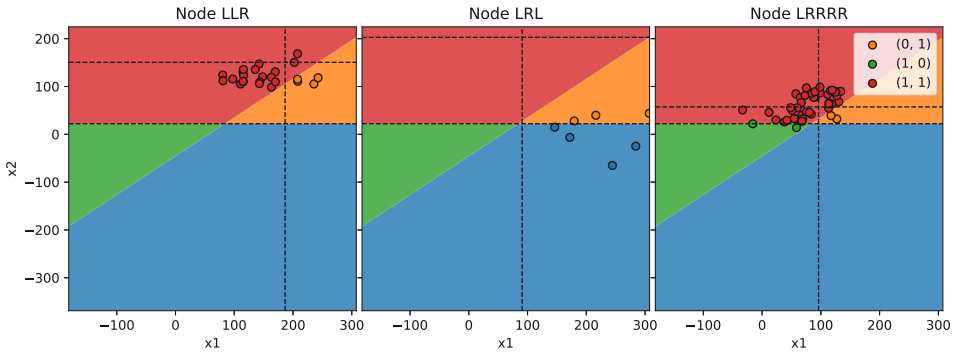


Figure 5.5: Scatter plot of the instances of the interpreted nodes explaining difference class $(0, 1)$, "One Classifier Ignores a Feature" example.

5.2.6 Interpretation Order

Each feature needs separate dependence plots. For the artificial examples in this thesis, which only have two features, this is no problem. But in which order should they be investigated, if there are more features? One could use one of the individual classifier's feature importances as calculated with the algorithm described in Section 2.4.5 to determine an order. But an ordering based on relevance for interpreting differences might be preferable. We can derive a feature importance from the SHAP Value differences in the same manner and use that.

5.3 Testable Design Proposition

To automatically derive the quantifiable quality metrics, I need to narrow down their definition.

5.3.1 Fidelity

Fidelity measures, how well the explanation is able to predict the model's behaviour. In each experiment, I use a separate test data set. Any of the performance measures can be used, like Accuracy, F1 Score, Precision and Recall. I select F1 Score, because it is a balanced version of Precision and Recall and is naturally calculated per class [Pow08]. This allows me to evaluate Fidelity for each class separately.

DiRo2C internally uses an intermediate surrogate model, a decision tree, from which it derives rules that are output for interpretation of the user. This tree can be directly used to evaluate against the model to be explained.

Because DiRo2C in its current implementation only supports explaining the binary difference classifier when multiclass classifiers are compared, I break down the explanation task into multiple tasks by reformulating the multiclass difference classification task for DiRo2C in a one-vs-rest manner. So the first task is to explain class $(0, 1)$ vs. all other classes, the second to explain class $(1, 0)$ vs. all other classes and so on.

The surrogate decision tree as an intermediate result of each of these explanations can be directly used to predict new instances. Trees with varying complexity are then obtained by pruning the full tree one step at a time using cost complexity pruning, with the implementation provided by scikit-learn [PVG+11] based on the algorithm of Breiman et al. [BFOS84, p. 66]. The trivial tree, consisting only of one node, is removed. They are sorted by their number of leaf nodes in descending order.

As a baseline, I choose surrogate decision rules derived from a decision tree which has been trained on the explanation data set to predict the difference classifier labels. It combines the widely used interpretability method of surrogate decision rules and Staufer and Rauber's notion of the difference classifier. But in contrast to DiRo2C, no new instances are generated for training the surrogate decision tree. Also, I am not creating an explanation per difference classifier class in a one-vs-rest manner, but just one to

explain all classes which can be evaluated in one go. This is because there is not the same limitation as with DiRo2C not supporting comparison of multiclass classifiers. To derive explanations with different complexities, the same pruning technique as described in the last paragraph is used.

Mocca-SHAP does not support predicting unseen instances out-of-the-box. Instead it is a visual, interactive framework. But I build a proxy task around it, that allows to predict unseen instances and captures the core idea of the modular, hierarchically clustered explanations.

At first, the clustering structure is obtained. Then, counterfactuals are calculated for each cluster node and for all classes of the difference classifier. Because counterfactuals denote upper and lower boundaries for a feature, I can derive a decision rule from each.

Consider an example where we want to do this for instances in a cluster that are classified 0 by A and 1 by B. If the lower counterfactual explanation is $x_1 = 0 \rightarrow (0, 0)$ and the upper counterfactual $x_1 = 100 \rightarrow (1, 1)$, then the range between them can be described with the decision rule $0 < x_1 < 100$. It describes the range, in which the focus instances are classified differently.

There might be other features with counterfactuals in this cluster node as well. In that case, they are combined with a logical *and* ($\wedge$). This results in a single decision rule for the cluster, which explains why these instances are classified differently. Now we have it in the same form as in the baseline and DiRo2C.

On the global level, we now get one decision rule per difference classifier class. But which decision rule now applies to a new instance? This can be solved with explanation similarity. First, generate the SHAP Values for the new instance. Then, find the instance of the data set with the most similar explanation and get that one's decision rule. Use this decision rule to predict the new instance's difference classifier class.

How do I evaluate on the local levels? We can obtain slices through the clustering hierarchy, where each slice contains clusters, such that all instances of the data set are included once in a slice. We start with the first slice, including node L and node R, the two child nodes of the root node. Then, we replace the node with the biggest cluster distance with its child nodes, and use that as the resulting set of cluster nodes as the next slice. This process is repeated, until we arrive at the leaf nodes. The cluster distance is a measure that represents the heterogeneity within a cluster. So in each step, we replace the most heterogeneous cluster with more specific ones.

5.3.2 Complexity

Because now all approaches to be compared include decision rules, we have a common basis on which to compare complexity. It is measured as the number of constraints that the decision rules are made of.

5.3.3 Generation Time

Because of the long generation time, two machines are used to run the experiments. Therefore, care needs to be taken when comparing generation time. The default machine is an Apple MacBook Pro with an Intel® Core™ i5-4278U CPU @ 2.60GHz. The aiding machine is an Amazon EC2 instance with an Intel® Xeon® CPU E5-2676 v3 @ 2.40GHz and was used to generate the DiRo2C explanations for the benchmark experiments.

5.4 Implementation

5.4.1 Programming Language and Libraries

I choose *Python* as the programming language, with which I implement the tools needed for running the experiments, and *Jupyter Notebooks* to set up the experiments and create the visualizations. The reason being, that I am already familiar with them and because Lundberg also published his package `shap` for *Python* [LL17]. The machine learning library `scikit-learn` [PVG+11] is used, because it includes many classifier types and has been used by Staufer and Rauber for evaluating DiRo2C [Sta21], of which some explanation tasks are used in this work. Another library, `XGBoost` [CG16], is used for training ensemble classifiers for the benchmark experiments. For data handling, transformation and visualization, the packages `numpy`[1], `scipy` [VGO+20] and `pandas`[2] are used. For creating plots, `matplotlib` [Hun07] and `seaborn` [Was21] are used.

5.4.2 Reproducibility

All data set splits for train and test data are done with a random number generator where a fixed initialization seed is set to be reproducible across runs. The prototypically implemented tools and the notebooks used to conduct the experiments are shared in a git repository[3]. A `README.md` file is also included, which contains information about how to set up a Python environment with the same packages and package versions.

5.4.3 Surrogate Decision Rules

The interpretability method of surrogate decision rules is used in the baseline and DiRo2C. To be able to interpret the most important rules first, they are sorted by support. Support measures the number of training instances covered with the rule. Within a rule the constraints are sorted by the feature's importance, as calculated from the surrogate decision tree. The implementation used offers importance measures as the normalized total reduction of the Gini criterion brought by each feature [sld]. The complexity of decision rules is calculated by counting all constraints, or more precisely, by the total number of relational operators.

[1] numpy: `https://github.com/numpy/numpy`
[2] pandas: `https://github.com/pandas-dev/pandas/`
[3] mocca-shap: `https://github.com/karltm/mocca-shap`

5.4.4 Mocca-SHAP

Log of Odds Transformation

In case the log of odds outcomes are chosen to be explained with Mocca-SHAP, simply applying the *logit* function to transform probabilities may introduce larger rounding errors than computing them from values in log-probability space. If probabilities are very small, then the log-probability space introduces less errors because of the way, that numbers are represented in computers. Furthermore, many scikit-learn classifiers directly support log-probability prediction. Instead of division that is done in probability space to obtain the odds ratio, subtraction is used in log-space. See Equation 5.2 for the formula. $log(p)$ is a direct output of the difference classifier. $log(1-p)$ is the inverse log-probability, which can be calculated by adding up all other classes' outcomes in probability space. Fortunately, the package `scipy` provides an optimized function with `logsumexp`[4] for that.

$$logit(p) = log(\frac{p}{1-p}) = log(p) - log(1-p) \tag{5.2}$$

Visualizations

In SHAP dependence plots, it may happen that interesting instances cannot be seen anymore because they are overridden by other instances. To avoid this, I implement that the focus instances are plot in front of all other instances. Focus instances are all instances that have a difference classifier label equal to the current class in focus. If no class is in focus, all instances that have a difference class label are in focus, because they generally are of a lower number than those that have an equality class label and I want to focus on the differences.

Clustering Algorithm

For clustering the difference classifier SHAP Values, I am using the same algorithm as Lundberg in his proposed *supervised clustering* approach [LEL19]. He used Euclidian distances and hierarchical clustering with complete linkage, as implemented in the package `scipy` [VGO+20]. To make use of all explanation information, the three-dimensional SHAP Values need to be transformed from a dimension of $n \times p \times q$ to $n \times pq$ for n instances, p features and q classes of the difference classifier.

Clusters in the cluster hierarchy are referred to as cluster nodes or just nodes, and are named according to their relative position to the root: For each link in the path down to a node it receives the letter L if it is left and R if it is right. A convenience function `descend` has been implemented to be able to quickly get to the lowest node, that still contains the same focus instances.

[4] scipy.special.logsumexp: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.logsumexp.html`, accessed: 2022-05-03

The proxy that enables predicting new instances is based on SHAP Value similarity. This is implemented with scikit-learn's K-Nearest Neighbors (KNN) classifier. By setting the number of neighbors to one, it predicts the cluster with the most similar explanation.

5.5 Summary

In this section I have described how I developed Mocca-SHAP incrementally and how the findings during the experiments contributed to that. I have answered research question #2 by extending the notion of the difference classifier for probability support, developed further variants and found out, that only the multiclass difference classifier is suited for use with SHAP Values. Research Question #3 was answered by selecting SHAP Dependence Plots as a suitable global extension method, which also support explaining modular parts of the explanation. I have also described how I arrived at the side-by-side SHAP Dependence Plots explaining the individual classifiers and the additional difference SHAP Values. I have shown various approaches to feature importances, and have proposed *group counterfactual explanations*, which are included in SHAP Dependence Plots when a focus class is set. They enable causal interpretations, which otherwise would be hard to interpret from the plots alone. To answer research question #4, I have proposed a clustering approach, that makes it possible to interpret parts of the explanation in a modular way based on similarity of the SHAP Values. Together with the group counterfactual explanations, this allows me to define a proxy that can be used to explain new instances and automatically derive quality metrics. In the last section, I have shared implementation details and how the results can be reproduced or used to conduct further research.

CHAPTER 6

Experiments

In this section, I describe the experiments that have been conducted to demonstrate Mocca-SHAP's basic working and assess on a range of automatically measured and qualitatively described quality metrics in comparison to a baseline and DiRo2C.

6.1 Running Example

This initial example serves to demonstrate how the methods basically work. The classifiers A and B predict differently in three areas of the feature space, with one area having no instances in the explanation data set. All values shown in decision rules are rounded to one decimal for display. These precisions are also used as the step sizes in the Mocca-SHAP algorithm for generating group counterfactual explanations. Typically, features in explanations would be sorted in descending order by their importances, but this has been overridden with the default order to ease comparison.

6.1.1 Baseline

Training the surrogate decision tree takes less than a second and results in a tree with a depth of three and eleven nodes. To be able to interpret explanations with different complexities, I prune it back one node at a time until the minimal depth of one is reached. This yields four trees, which I interpret in descending order by complexity. In each explanation I visualize the tree and interpret the derived decision rules and watch them evolve. In the last explanation, I arrive at the full tree again, which seems to be the most useful because it is still simple enough to be interpretable and predicts all focus instances correctly. The derived decision rules for the difference classes are listed below and a plot of their decision boundaries is shown in Figure 6.1.

1. if $-25.5 < x_1 \leq 148.0 \wedge x_2 > -140.1$, then $(1, 0)$

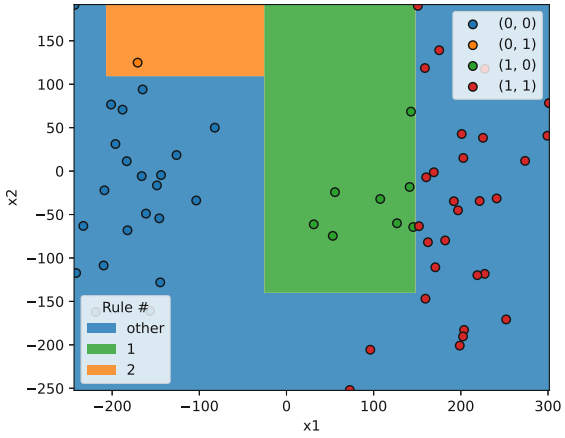


Figure 6.1: Decision boundaries of the fourth baseline explanation, running example. The instances shown are from the explanation data set and are colored by their difference classifier label.

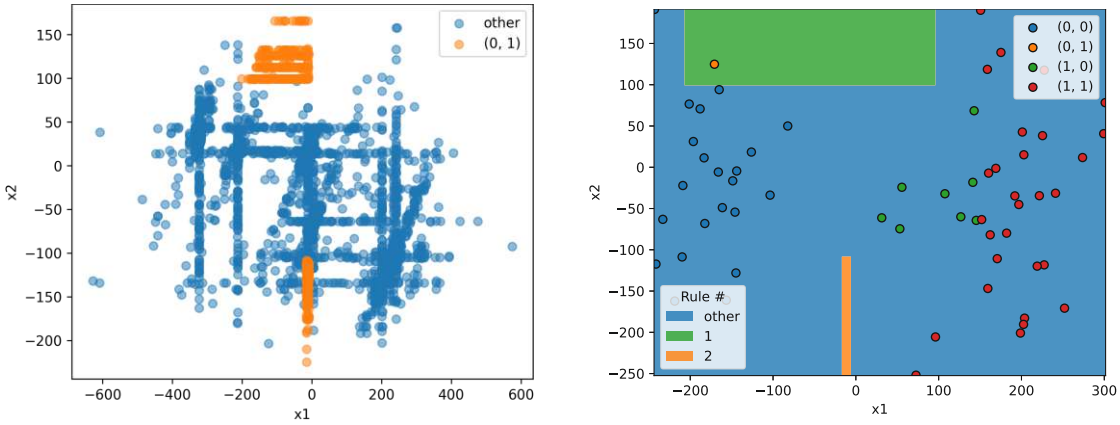
2. if $-207.0 < x_1 \leq -25.5 \wedge x_2 > 109.4$, then $(0, 1)$
3. else, both classifiers predict the same label

This explanation achieves a fidelity of 79% F1 score for class $(0, 1)$ with three constraints and 80% F1 score for class $(1, 0)$ also with three constraints. When comparing the boundaries to the ground truth, I notice that the upper x_2 boundary is missing in the second decision rule. There are no training instances in this area, so the decision tree was not able to know how this area should be classified. Also, a decision rule to describe the smaller $(0, 1)$ differences for low x_2 values is missing completely. There are also no instances in the area from which the surrogate model could have learned the correct class.

6.1.2 DiRo2C

In contrast to the baseline, DiRo2C generates one explanation per class of the difference classifier. In total, generation takes approx. seven minutes. The explanation with class $(0, 1)$ in focus includes a tree, which has a depth of four and includes 13 nodes. In the same manner as in the baseline approach, pruning yields now three trees with different complexities. The last explanation also explains the single instance of the explanation data set with the label $(0, 1)$ and includes a decision rule which does not apply to any instances in the explanation data set. The decision rules are listed below. Figure 6.2 shows the decision boundaries of the explanation and a scatter plot of DiRo2C's generated instances.

1. if $-207.0 < x_1 \leq 96.2 \wedge x_2 > 99.1$, then $(0, 1)$
2. if $-16.7 < x_1 \leq -5.9 \wedge x_2 \leq -108.3$, then $(0, 1)$



(a) Scatter plot of the generated instances.

(b) Decision boundaries of the decision rules.

Figure 6.2: Third DiRo2C's explanation for difference class $(0, 1)$, running example.

3. else, then not $(0, 1)$

DiRo2C performs worse for class $(0, 1)$ than the baseline, it achieves only 71% with six constraints. You can see in Figure 6.2a, that the upper area contains only a few generated instances and thus the surrogate tree is not able to learn the correct boundaries for this class. But in contrast to the baseline, it did describe the second area on the bottom with decision rule 2.

The explanation with class $(1, 0)$ in focus includes a tree with a depth of four and offers four explanations with different complexities after pruning. At the last explanation, all instances of the explanation data set with the label $(1, 0)$ are explained. There is just one decision rule, shown below. Its decision boundaries and a scatter plot of the generated instances are shown in Figure 6.3.

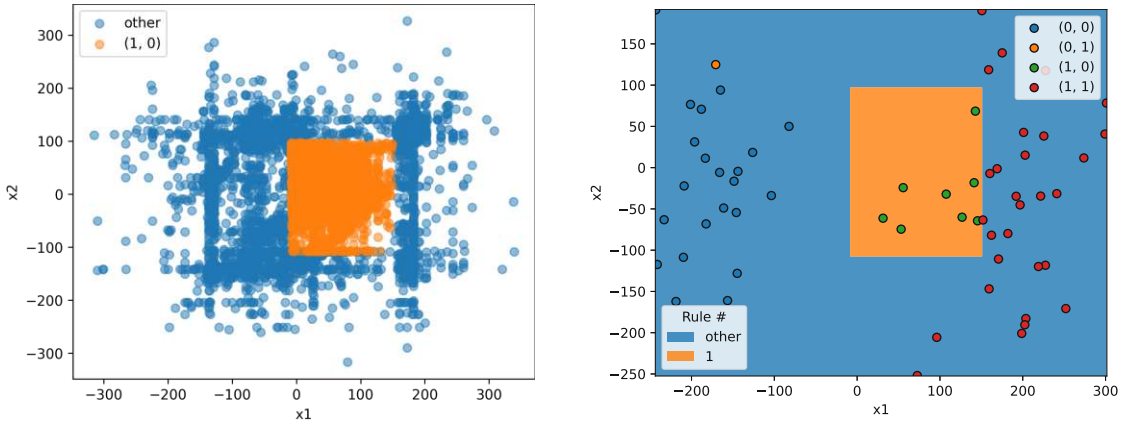
1. if $-8.5 < x_1 \leq 151.1 \wedge -107.9 < x_2 \leq 97.4$, then $(1, 0)$
2. else, then not $(1, 0)$

For class $(1, 0)$, it achieved 100% F1 score with four constraints.

6.1.3 Mocca-SHAP

I choose to explain the probability space with Mocca-SHAP. But the classifiers estimate only hard probabilities, which means they either predict 0% or 100% for a class. Because of this, the exposed probability estimates are equal to the one-hot encoded labels. Generation takes approx. 20 seconds.

I start by visualizing the feature importances of the individual classifiers, shown in Figure 6.4. Feature x_2 is of no importance to A. This is because of the *missingness* property of



(a) Scatter plot of the generated instances.

(b) Decision boundaries of the decision rules.

Figure 6.3: Fourth DiRo2C explanation for difference class $(1, 0)$, running example.

SHAP and the fact that the mean absolute SHAP Value of x_2 is exactly 0. So while A bases its decisions solely on x_1, B uses both features.

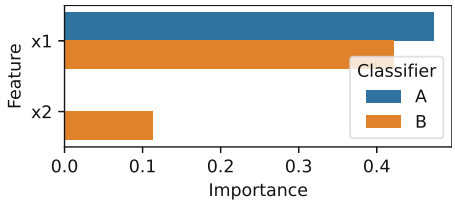


Figure 6.4: Individual classifiers' feature importances, running example.

The constant effect of x_2 on A's probability estimate of class 1 can be seen clearly in the individual SHAP Dependence Plots in Figure 6.5, because all SHAP Values are in a horizontal line at 0. Note, that only the effect on class 1's outcome is shown, because for binary classifiers class 0's outcome is just the inverse and thus includes no additional information. x_1's dependence plot for A is in the shape of a step, having a constantly negative effect up to a certain point and a constantly positive effect above. In contrast, classifier B's dependence curves are more complex, with multiple steps and vertical dispersion, which results from feature interactions. It depends on the combination of feature values of x_1 and x_2 for B to decide which class to assign to an instance. For low values of x_2 (below approx. -100), both classifiers agree on assigning either class 0 or class 1, i.e. in this range their decision boundaries are identical. For values of x_2 between approx. -100 and 100, the classifiers either agree on class 1 or class 0 (red and green dots), but there are also some where they disagree, with classifier A assigning class 1 and classifier B assigning class 0 (orange dots). These are instances that have higher values for x_1 as can be seen in the first row middle figure. For higher values of x_2 above approx. 100, the classifiers again agree on class 0 or 1, save for one instance (blue dot) that has a low value for x_1.

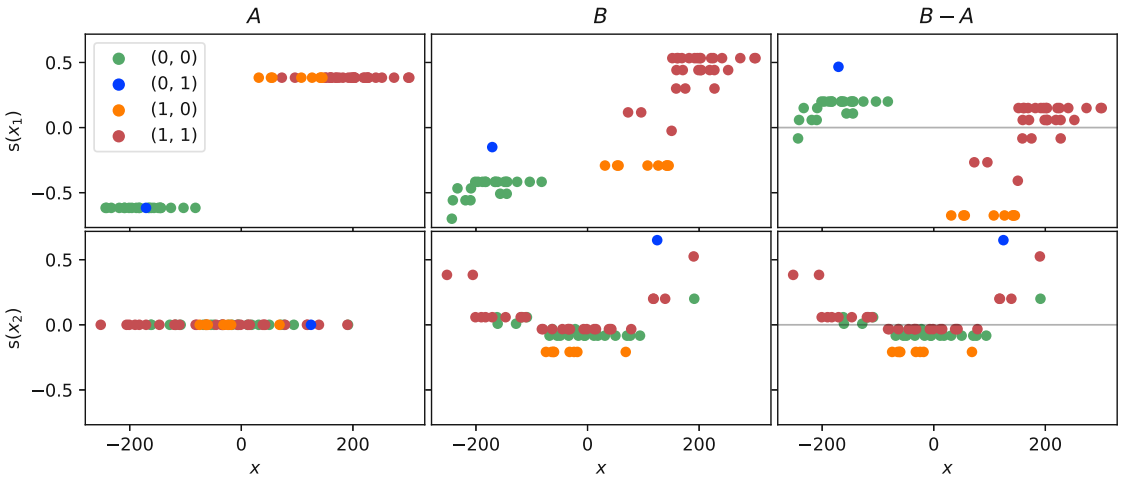


Figure 6.5: Individual and difference dependence plots, running example. The first row shows x_1's effects, the second x_2's effects on class 1's outcome.

I continue by interpreting dependence plots for SHAP Value differences, shown in the right column. They have the advantage, that I can interpret over- and underestimation of classifier B. B overestimates the outcome for low x_1 values (below approx. 0). In one case (blue dot) this overestimation is especially big, which could explain why B predicts class 1 in this case instead of class 0 like A. Furthermore, B underestimates the outcome for medium values of x_1 between approx. 0 and 100. This could explain why certain instances are classified $(1, 0)$ (orange). For low x_2 values it overestimates the outcome (below approx. -200), but without any instances classified differently. For medium values it underestimates the outcome (from approx. -200 until 100) in certain cases (orange dots), which could explain why they are classified 0 by B instead of 1 like A. For high values it overestimates the outcome (above approx. 100), where there is one instance with peak overestimation (blue dot), which could explain why it is classified 1 by B and 0 by A.

Difference classifier dependence plots, including group counterfactual explanations, allow interpretation with causal links. In Figure 6.6 I have created them for x_1 with class $(0, 1)$ in focus and in comparison with the effects on class $(0, 0)$ and $(1, 1)$'s outcomes. For brevity, class $(1, 0)$ is excluded from the plots. x_1's effect on class $(0, 1)$'s outcome is marginally higher in between the counterfactual boundaries. Thus, the probability predicted for class $(0, 1)$ of these instances is slightly higher. But the instance classified $(0, 1)$ has an especially high SHAP Value. To investigate the behaviour in this range in more detail, I extract a subset with all instances in the range (-203.8, -8.5) and call it node α.

Next, I investigate x_2's effects for instances part of node α and those not part of it separately, as shown in Figure 6.7. Now I can see, that the magnitude of the step-shaped relation depends on x_1: if the instances are part of node α, the increase in $(0, 1)$'s outcome

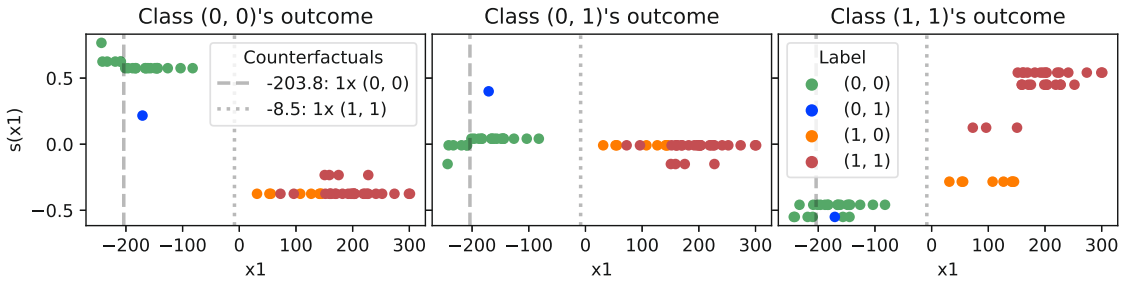


Figure 6.6: Difference classifier dependence plots for x_1 with class $(0, 1)$ in focus, running example.

for $x_2 > 97.5$ is much bigger than for instances not part of it. Also, node α only contains instances which A classifies as 0 and B as either 0 or 1, while the other node $\neg\alpha$ contains mainly instances that A classifies as 0 and B as 0 or 1.

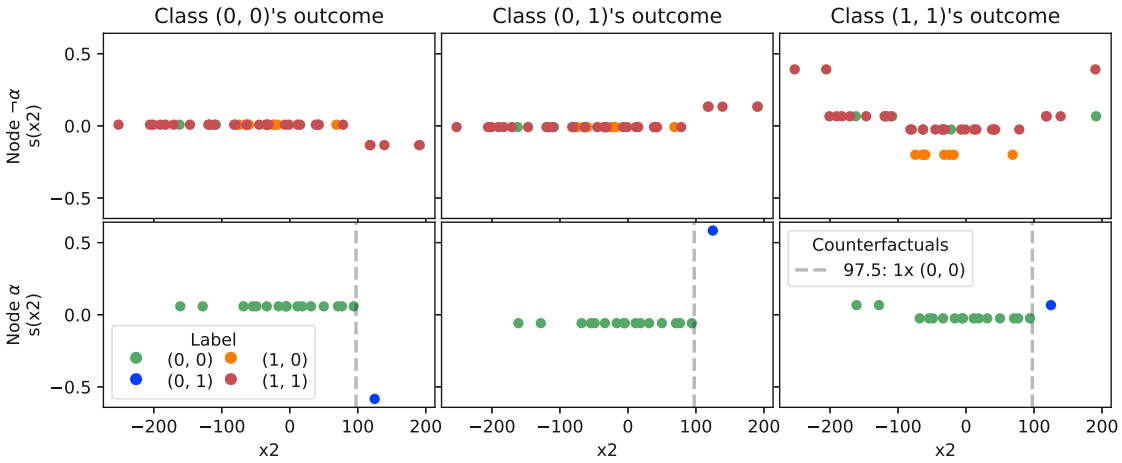


Figure 6.7: Difference classifier dependence plots for node α and all instances not part of node α with class $(0, 1)$ in focus, running example.

Mocca-SHAP achieves the highest overall fidelity for class $(0, 1)$ with 97% compared to the other approaches, although it did not find the the second area classified differently for low x_2 values.

I continue by putting class $(1, 0)$ into focus. The dependence plots for x_1 are shown in Figure 6.8. This time, class $(0, 1)$ is omitted for brevity. The effect of x_1 on class $(1, 0)$'s outcome is constant and marginally lower than 0 for low feature values and high feature values, but higher for medium values as can be seen in the middle figure. To investigate further, I extract instances between the two counterfactual boundaries in the range (-8.6, 150.8), and call the new node β.

In Figure 6.9 you can see dependence plots for x_2, with instances part of node β in the second row and those not part of β in the first row for reference. Most of the vertical

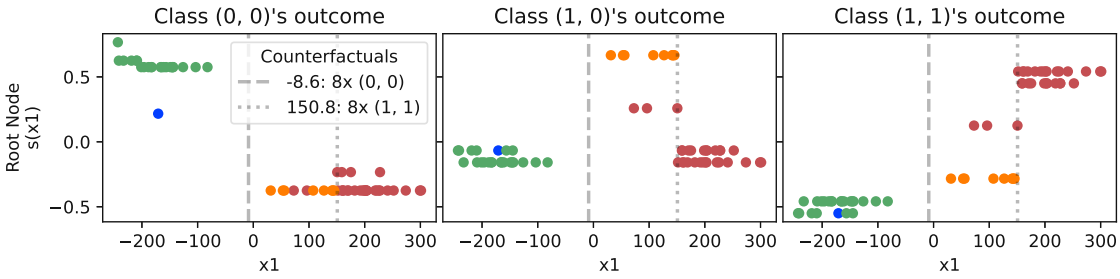


Figure 6.8: Difference classifier dependence plots for x_1 with class $(1, 0)$ in focus, running example.

dispersion previously observed breaks down to simpler step-like relations. Only between the two counterfactual explanations in the range (-108.2, 97.6), instances in node β are classified $(1, 0)$, and have SHAP Values with greater magnitude than the instances not in β. This has an increasing effect on the probability of $(1, 0)$. Also, in the right plot of node β you can see, that x_2 has a bigger decreasing effect on the probability of $(1, 1)$ for these instances. This distinguishes them from the red instances, which are classified $(1, 1)$. They are the only other instances in node β.

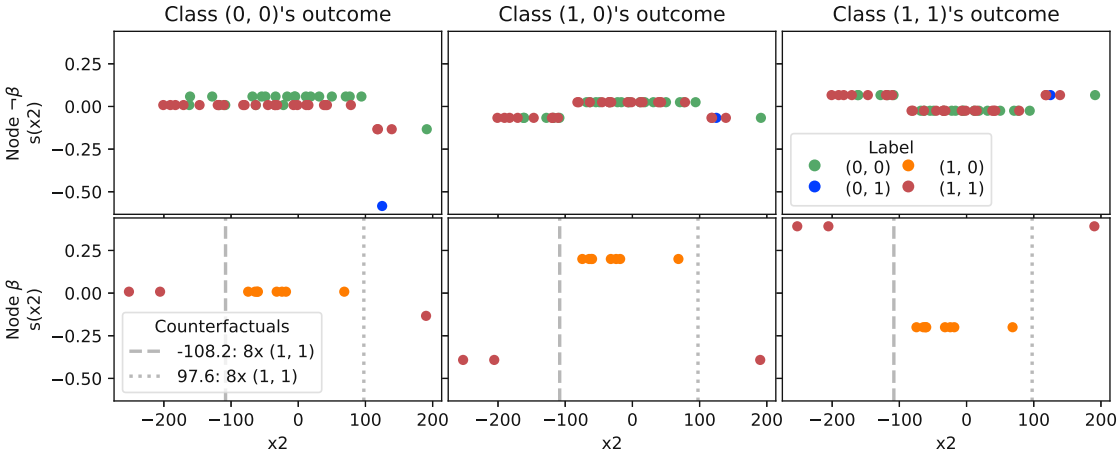


Figure 6.9: Difference classifier dependence plots for node β and all instances not part of node β with class $(1, 0)$ in focus, running example.

I conclude, that B predicts 1 while A predicts 0 for instances having $-203.8 < x_1 \leq -8.6 \wedge x_2 > 97.5$ and, that B predicts 0 while A predicts 1 for instances having $-8.6 < x_1 \leq 150.7 \wedge -108.2 < x_2 \leq 97.5$. In Figure 6.10 you can see a scatter plot of the instances in nodes α and β.

Mocca-SHAP achieves 100% with four constraints for class $(1, 0)$, just like DiRo2C.

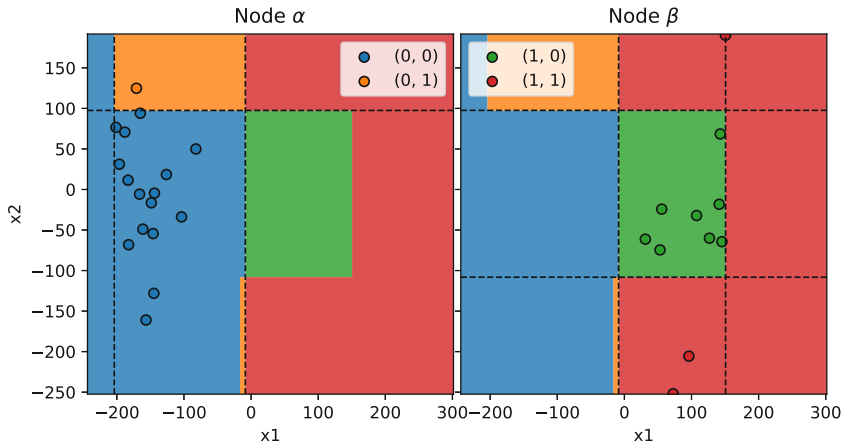


Figure 6.10: Scatter plots for instances of nodes α and β and their counterfactual boundaries (dashed lines), running example.

6.1.4 Comparison

You can further see a plot of the achieved F1 scores for explanations with different complexities as automatically calculated in Figure 6.11. In the automatic runs, not only explanations for the two difference classes $(0, 1)$ and $(1, 0)$ are evaluated, but also for the equality classes $(0, 0)$ and $(1, 1)$. Counterfactuals can be computed in the same way for equality classes, and denote the feature value, which the instances would have to assume for the difference classifier class to change.

- Explanations for class $(0, 0)$: Both DiRo2C and Mocca-SHAP perform very similarly over different degrees of complexity, and both achieve higher F1 scores than the baseline.
- Explanations for class $(0, 1)$: Note, that both Mocca-SHAP and the baseline generated only one explanation, because of the simplicity of the task. Therefore, there is no curve, but only dots in the graph. They both achieved higher fidelity than DiRo2C, with Mocca-SHAP scoring highest.
- Explanations for class $(1, 0)$: Note, that Mocca-SHAP generated only one explanation because of the simplicity of the task. At this level of complexity (four constraints), both Mocca-SHAP and DiRo2C achieve the highest fidelity, which is higher than the maximally achieved fidelity of the baseline with a complexity of three.
- Explanations for class $(1, 1)$: The baseline achieves only low fidelity and does not offer more complex explanations like Mocca-SHAP and DiRo2C. Mocca-SHAP achieves a marginally higher fidelity than DiRo2C.

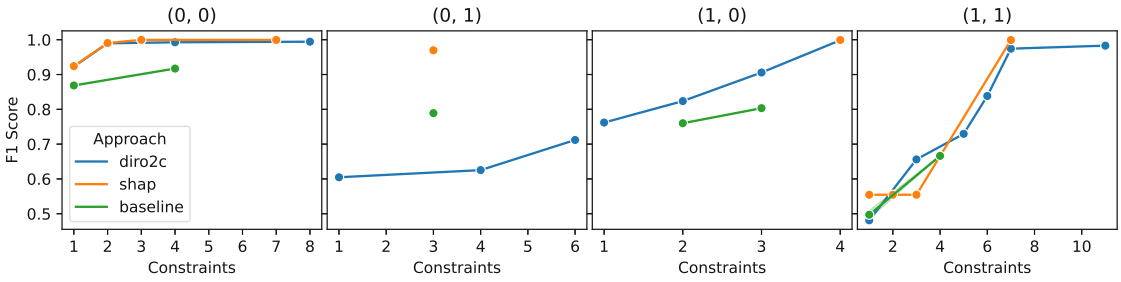


Figure 6.11: Explanation fidelity on the running example test set for explanations with different complexities.

Generation time was the longest for DiRo2C with seven minutes, followed by Mocca-SHAP with 20 seconds. The baseline generation finished nearly instantly.

6.1.5 Summary

Mocca-SHAP achieved highest fidelity in the automatic evaluation for all four difference classifier classes, while DiRo2C was on the same level in three of them. DiRo2C performed bad on the $(0, 1)$ explanations. This was an obstacle during interpretation as well, because it missed one boundary of the decision rule for difference class $(0, 1)$. Also, it took the longest to generate. The only advantage of DiRo2C was, that it was the only method which found a decision rule for the second part of the $(0, 1)$ differences with lower x_2 values, which does not include any examples in the data set.

6.2 One Classifier Ignores a Feature

In this artificial task, the goal is to explain one area classified $(0, 1)$ and another classified $(1, 0)$, each with one horizontal boundary and one tilted boundary, that is, it is not axis-aligned. All values shown in decision rules are rounded to one decimal for display. These precisions are also used as the step sizes in the Mocca-SHAP algorithm for generating group counterfactual explanations. Typically, features in explanations would be sorted in descending order by their importances, but this has been overridden with the default order to ease comparison.

6.2.1 Baseline

The baseline approach generated a surrogate decision tree with a depth of four, which, if pruned down to minimal depth, offers five explanations to interpret. Generation time was less than a second. Explanation (1) does not describe the difference classes at all, (2) only has a coarse rule describing the $(0, 1)$ differences: $x_1 > 174.8 \wedge x_2 > 23.9$, (3) adds a rule describing the $(1, 0)$ differences: $x_1 \leq 69.8 \wedge -29.9 < x_2 \leq 23.9$, (4) adds an upper boundary to the first rule and (5) adds another rule describing $(0, 1)$. Finally,

the rule set explains every instance classified differently. They are listed below and a visualization of their decision boundaries is shown in Figure 6.12.

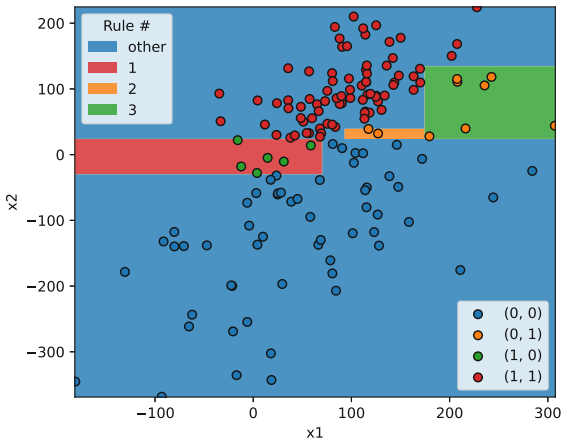


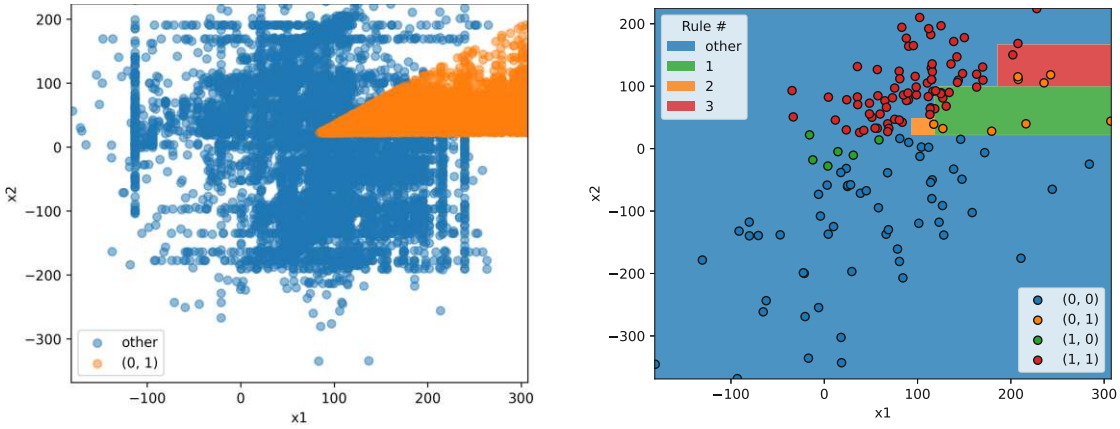
Figure 6.12: Decision boundaries of the 5th baseline explanation, "One Classifier Ignores a Feature" example. The instances shown are from the explanation data set and are colored by their difference classifier label.

1. if $x_1 > 174.8 \wedge 23.9 < x_2 \leq 134.3$, then $(0, 1)$
2. if $x_1 \leq 69.8 \wedge -29.9 < x_2 \leq 23.9$, then $(1, 0)$
3. if $93.3 < x_1 \leq 174.8 \wedge 23.9 < x_2 \leq 39.4$, then $(0, 1)$
4. else, both classifiers predict the same label

This explanation achieves a F1 score of 81.5% for class $(0, 1)$ with seven constraints. The difference area is approximated with two rectangular shapes, one with its right side open, as can be seen in Figure 6.12. On class $(1, 0)$, this explanation achieves a F1 score of 73.3% with three constraints. The difference area is approximated with only one rectangular shape, which is open to the left.

6.2.2 DiRo2C

In total, generation took approx. 14 minutes. I start by interpreting the explanation for difference class $(0, 1)$. A scatter plot of the generated instances is shown in Figure 6.13a. The surrogate tree has a depth of 12. Pruning it back to the minimal tree yields 23 explanations. I choose the fifth explanation as the final explanation, because the more detailed ones don't lead to new insights and because every instance of the explanation data set which is classified $(0, 1)$ is explained. I derive the decision rules listed below. They are visualized in Figure 6.13b.



(a) Scatter plot of the generated instances.

(b) Decision boundaries of the decision rules.

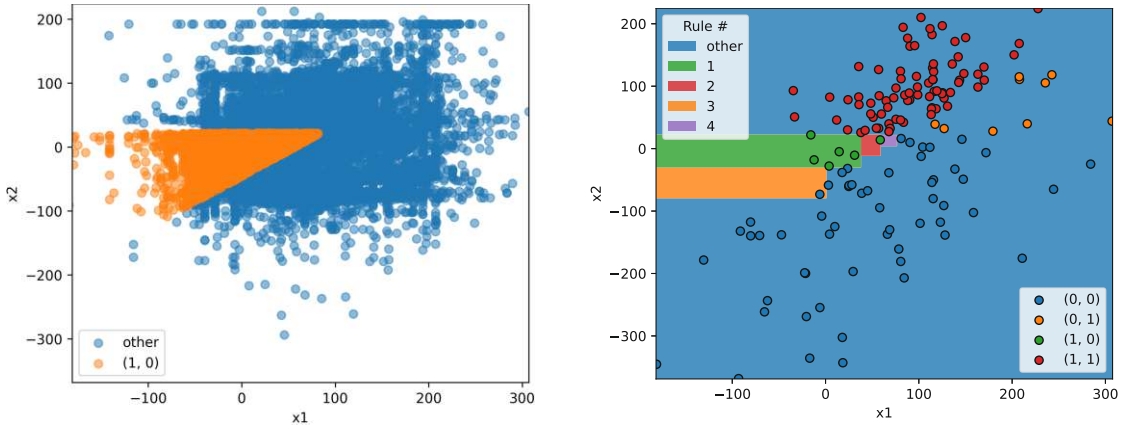
Figure 6.13: Last interpreted explanation of DiRo2C's explanation for difference class $(0, 1)$, "One Classifier Ignores a Feature" example.

1. if $x_1 > 118.5 \wedge 21.9 < x_2 \leq 99.8$, then $(0, 1)$
2. if $93.7 < x_1 \leq 118.5 \wedge 22.0 < x_2 \leq 48.8$, then $(0, 1)$
3. if $x_1 > 185.8 \wedge 99.8 < x_2 \leq 166.9$, then $(0, 1)$
4. else, then not $(0, 1)$

This explanation achieves a F1 score of 87% and thus scores higher than the baseline. It is also more complex with ten constraints, but still interpretable. Seen from the decision boundary plot in Figure 6.13a, the differences are is described with three boxes, of which two are open to the right. This is a more accurate approximation compared to the baseline.

I continue by interpreting the explanation for difference class $(1, 0)$. The surrogate tree has a depth of 13 and offers 48 explanations to interpret after pruning. I choose the seventh explanation, as the further ones do not lead to new insights. The decision rules of that explanation are listed below and a plot of their decision boundaries is shown in Figure 6.14.

1. if $x_1 \leq 38.2 \wedge -29.9 < x_2 \leq 22.1$, then $(1, 0)$
2. if $38.2 < x_1 \leq 59.3 \wedge -11.7 < x_2 \leq 22.1$, then $(1, 0)$
3. if $x_1 \leq 1.0 \wedge -79.7 < x_2 \leq -29.9$, then $(1, 0)$
4. if $59.3 < x_1 \leq 76.4 \wedge 3.9 < x_2 \leq 22.0$, then $(1, 0)$
5. else, then not $(1, 0)$



(a) Scatter plot of the generated instances.

(b) Decision boundaries of the decision rules.

Figure 6.14: Last interpreted explanation of DiRo2C's explanation for difference class $(1, 0)$, "One Classifier Ignores a Feature" example.

This explanation achieves a F1 score with 90%, which is higher than the baseline. It includes 14 constraints. As we can see in Figure 6.14a, the differences are is approximated with four rectangles, of which two are open on the left.

6.2.3 Mocca-SHAP

All preconditions are met for explaining the log of odds space: Both classifiers offer probability estimates and none of the estimated probabilities for the explanation data set is an extreme value of either 0% or 100%. With that, generation took approx. seven minutes. I start by interpreting the individual classifiers' feature importances, shown in Figure 6.15. Classifier B is different from A in that x_1 has no importance at all to it.

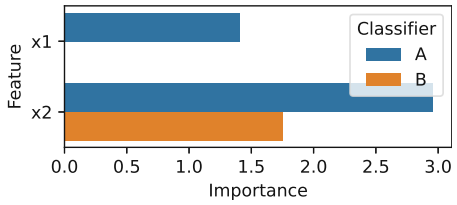


Figure 6.15: Individual classifiers' feature importances, "One Classifier Ignores a Feature" example.

This is confirmed in the individual dependence plots for the effects on class 1's outcome shown in Figure 6.16: x_1 has no effect on B's outcomes, because its effect is constantly 0. On A's class 1 outcome, it has a linearly decreasing effect. In contrast, x_2's effect is similar for both classifiers, which is linearly increasing, but less steep for B. According to the difference dependence plots in the right column, up to a certain point B underestimates the effect of x_1 on class 1's outcome compared to A, and overestimates the effect above.

Furthermore, it overestimates the effect of x_2 up to a certain point and underestimates its effect above. You can see that instances classified $(1, 0)$ (orange) are located below the horizontal line at $s = 0$ in x_1's difference dependence plot, and that the instances classified $(0, 1)$ are located above this line. Vice versa, this is the case for x_2's difference dependence plots, except that some instances classified $(1, 0)$ are above the line.

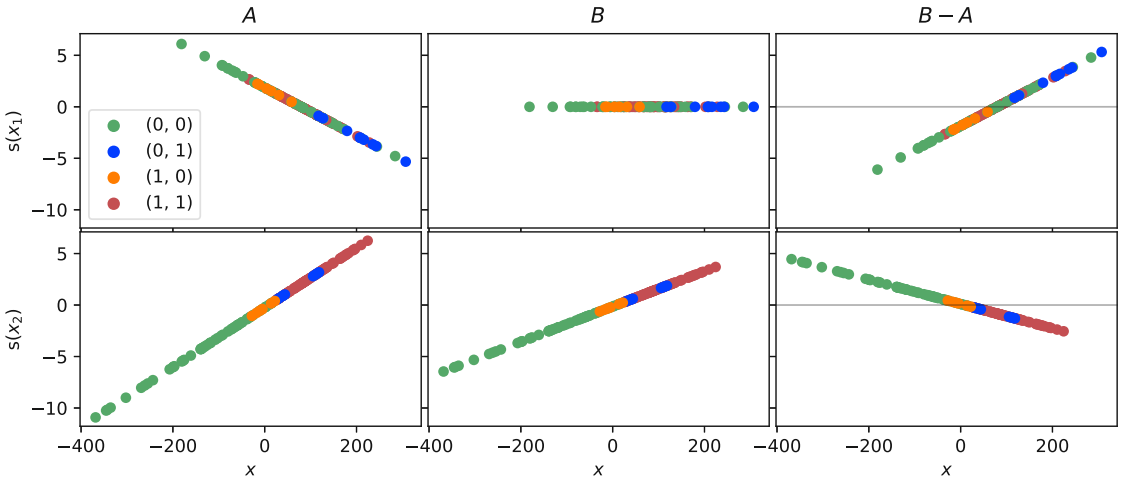


Figure 6.16: Individual and difference dependence plots, "One Classifier Ignores a Feature" example.

I continue by interpreting difference classifier dependence plots with a focus on the instances classified $(0, 1)$. I apply clustering as described in Section 5.2.5 to obtain clusters according to the difference classifier SHAP Values. The lowest node, that contains all instances classified $(0, 1)$ is L. Its child nodes LL and LR are of very different size (intra-cluster distance of 25 and 107), so I split the larger node (LR) again into LRL and LRR. Now I descend each cluster node to the point where their distances are about equal (intra-cluster distances of 25, 29 and 27), resulting in nodes LLR, LRL and LRRR. Their dependence plots are shown in Figure 6.17, along with plots for the global level. x_1's lower counterfactual varies for all interpreted nodes. x_2's upper counterfactual varies too, but its lower counterfactual is equal across all nodes.

I conclude, that generally B tends to estimate label 1 more often than A when $x_2 > 22.1 \wedge x_1 > 90.8$, because of the counterfactuals of x_1 and x_2 that do not vary across the interpreted clusters. But the upper boundary of x_2 and the lower boundary of x_1 depend on both features' values. Examples for such are shown below in the rules derived for the local cluster nodes. You can see scatter plots of each node's instances with the counterfactual explanations in Figure 6.18.

- *Node root*: if $x_1 > 90.8 \wedge 22.1 < x_2 \leq 202.7$, then $(0, 1)$
- *Node LLR*: if $x_1 > 186.6 \wedge 22.1 < x_2 \leq 150.5$, then $(0, 1)$

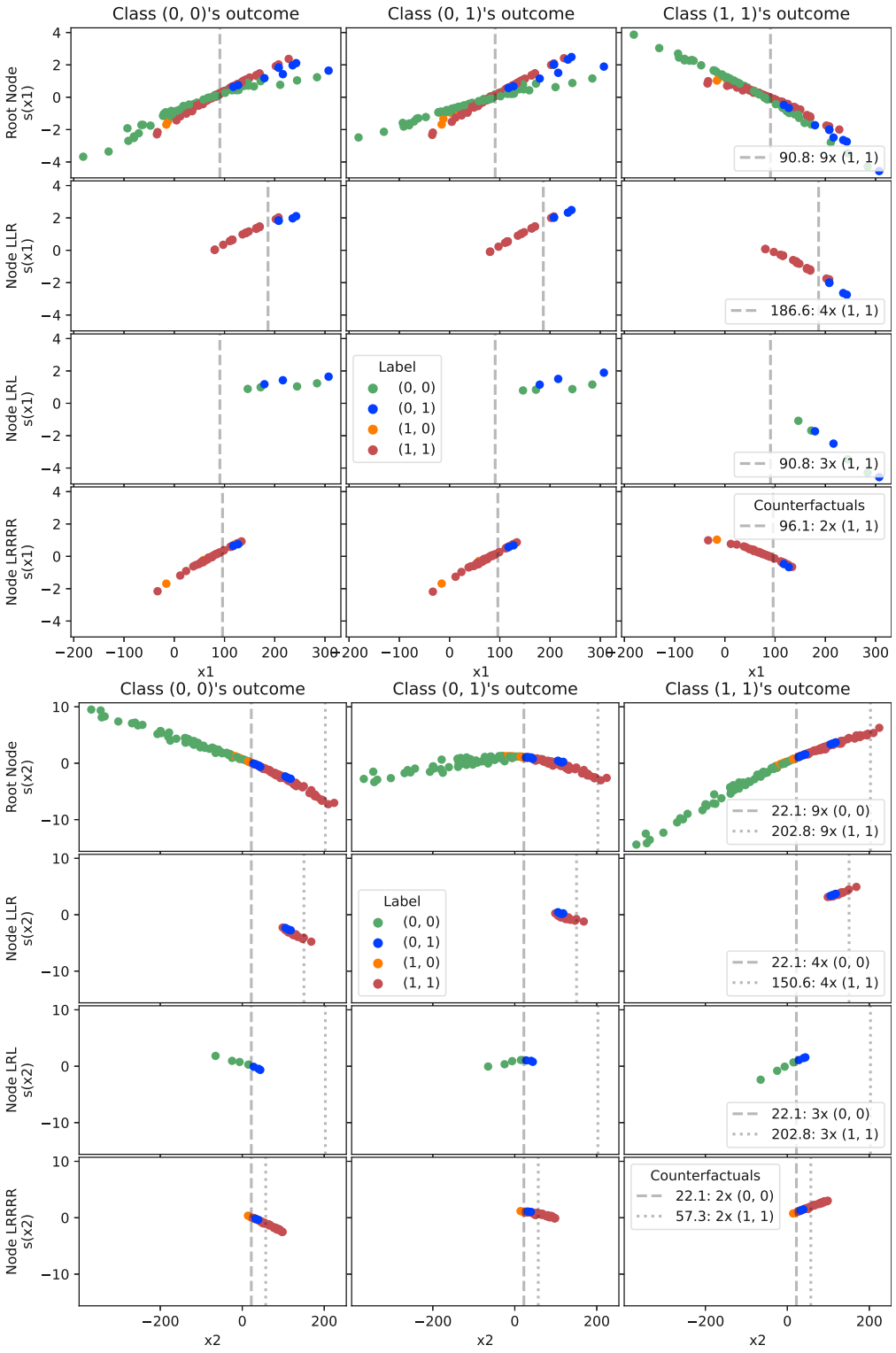


Figure 6.17: Difference classifier dependence plots with instances classified $(0, 1)$ in focus (blue), "One Classifier Ignores a Feature" example.

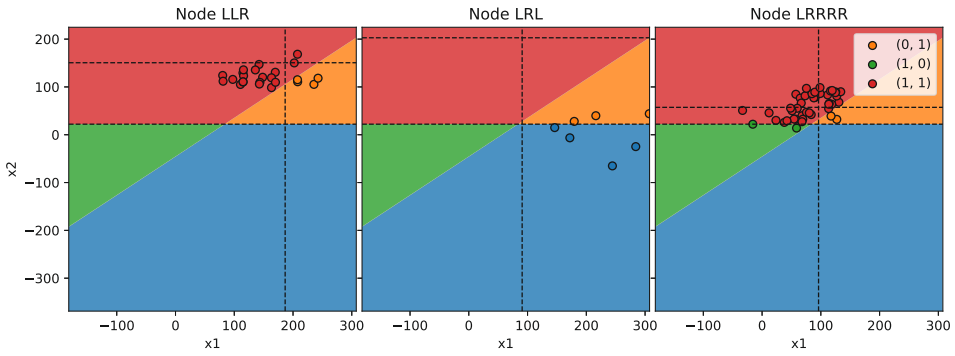


Figure 6.18: Scatter plot of the instances of the interpreted nodes explaining difference class $(0, 1)$, "One Classifier Ignores a Feature" example.

- *Node LRL*: if $x_1 > 90.8 \wedge 22.1 < x_2 \leq 202.7$, then $(0, 1)$
- *Node LRRRR*: if $x_1 > 96.1 \wedge 22.1 < x_2 \leq 57.2$, then $(0, 1)$

Mocca-SHAP achieves a marginally higher F1 score for class $(0, 1)$ with 88% than DiRo2C while also being less complex with nine constraints compared to ten constraints. From the plot including the counterfactual explanations for each cluster interpreted in Figure 6.18, you can see that nodes LLR and LRRRR nicely approximate the difference area. Node LRL's counterfactuals are close to the global counterfactuals, thus they do not explain the local behaviour so well.

Next, I investigate the difference classifier dependence plots with a focus on instances classified $(1, 0)$. One split of the focus instances seems sufficient, because the intra-cluster distances are quite similar with 25 and 34. Please note, that all instances classified $(1, 0)$ are contained in node LRR, so the first split assigns part of them into LRRL and part into LRRR. I create dependence plots for these nodes, shown along with the global level in Figure 6.20. Now there is a varying upper counterfactual boundary for x_1. x_2's lower counterfactual varies just slightly but its upper counterfactual is the same across all nodes.

I conclude, that B tends to estimate label 0 more often than A when $x_2 \leq 22.1 \wedge x_1 \leq 83.5$, because the corresponding counterfactuals stay the same across all clusters interpreted. But the lower boundary of x_2 and the upper boundary of x_1 depend on both features' values. Examples for such are shown below in the rules derived for the local cluster nodes. You can see scatter plots of each node's instances with the counterfactual explanations in Figure 6.19.

- *Node root*: if $x_1 \leq 83.5 \wedge -58.3 < x_2 \leq 22.1$, then $(1, 0)$
- *Node LRRL*: if $x_1 \leq 50.2 \wedge -55.5 < x_2 \leq 22.1$, then $(1, 0)$

- *Node LRRR*: if $x_1 \leq 83.5 \wedge -58.3 < x_2 \leq 22.1$, then $(1, 0)$

Mocca-SHAP achieves a lower F1 score with 82% that DiRo2C, but a higher score than the baseline. As can be seen in the plots of the nodes including the counterfactuals in Figure 6.19, the two local explanations are quite similar. Node LRRR could have been broken down once again to highlight local differences.

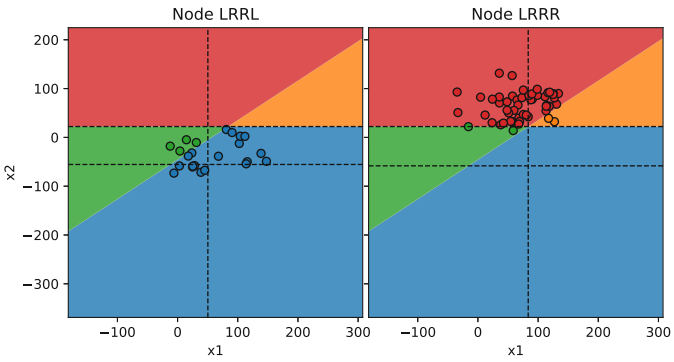


Figure 6.19: Scatter plot of the instances of the interpreted nodes explaining difference class $(1, 0)$, "One Classifier Ignores a Feature" example.

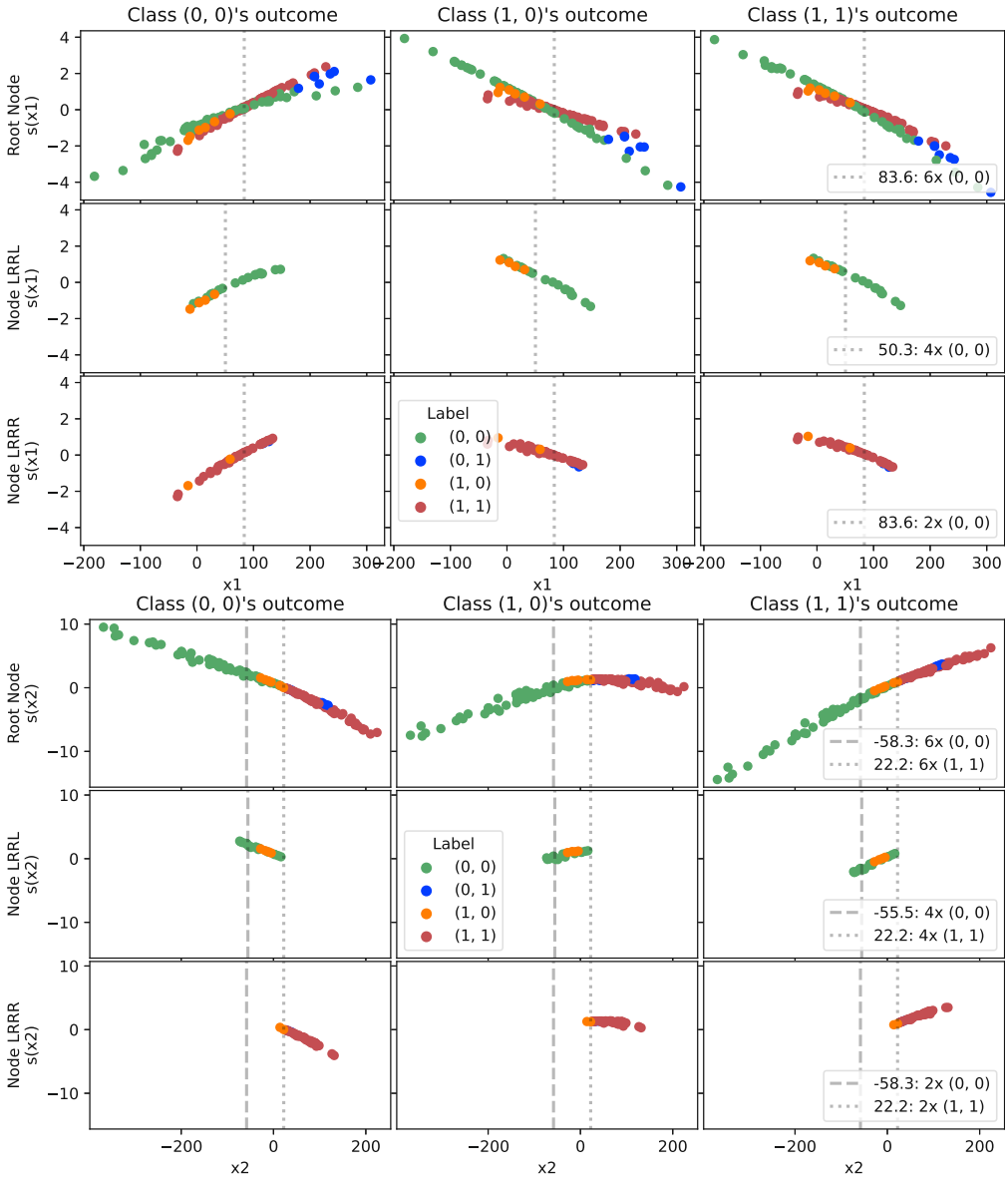


Figure 6.20: Difference classifier dependence plots with instances classified $(1, 0)$ in focus (orange), "One Classifier Ignores a Feature" example.

6.2.4 Comparison

As can be seen in Figure 6.21, Mocca-SHAP beats the baseline in terms of the highest achieved fidelity, but DiRo2C achieves even higher fidelity, at the cost of higher complexity. This shows a disadvantage of Mocca-SHAP: it can only be as accurate to the underlying classifiers, as the data set allows it. DiRo2C has an advantage, since it is able to generate more instances in sparse areas.

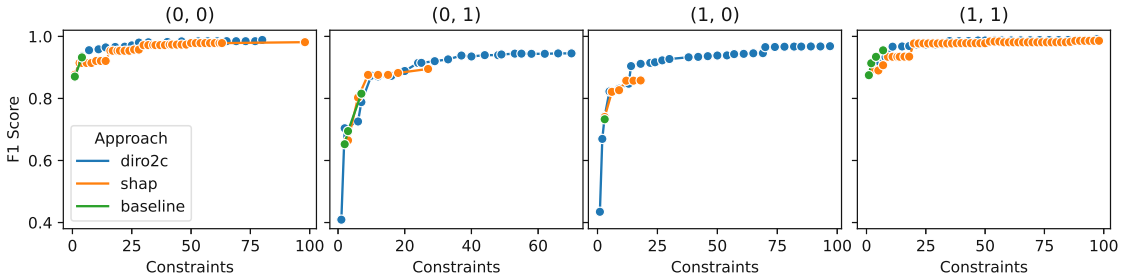


Figure 6.21: Explanation fidelity on the "One Classifier Ignores a Feature" example test set for explanations with complexities up to 100 constraints.

While all approaches allowed to interpret with a varying degree of detail, the baseline explanation was just enough to get a general understanding and offered no option to investigate in more detail. With DiRo2C it was possible to increase the complexity of the generated decision rules as desired, at the cost of strongly increasing complexity.

Only Mocca-SHAP allowed me to interpret that the $(0, 1)$ differences have a common, global lower decision boundary and the $(1, 0)$ differences a common, global upper decision boundary, and that the reason for this is that B ignores feature x_1.

DiRo2C explanation generation took the longest with approx. 14 minutes, followed by Mocca-SHAP with approx. seven minutes. The baseline generated the explanation nearly instantly.

6.2.5 Summary

Both Mocca-SHAP and DiRo2C achieve reasonable levels of fidelity in the automatic evaluation, but the baseline is restricted too much by its simplicity. DiRo2C even offers more complex explanations, but only at very high cost of complexity. When I interpreted the explanations, both Mocca-SHAP and DiRo2C allowed me to get a good understanding of the observed differences. But DiRo2C took the longest when generating its explanations.

6.3 Gaussian Quantiles

In this artificial task, the goal is to explain a ring-shaped area where classifier A predicts 1 and classifier B 0. There are no cases where A predicts 0 and B 1. All values shown

in decision rules are rounded to one decimal for display. These precisions are also used as the step sizes in the Mocca-SHAP algorithm for generating group counterfactual explanations. Typically, features in explanations would be sorted in descending order by their importances, but this has been overridden with the default order to ease comparison.

6.3.1 Baseline

The baseline approach generated a surrogate decision tree with a depth of eight, which, if pruned down to minimal depth, offers 23 explanations to interpret. Generation time was less than a second. In total, I interpreted the first 14 explanations. The last of them does not cover instances #124, #155, #169, #203 and #221 of the explanation data set, which are classified differently, but it would be possible to explain them with an explanation with higher complexity. The decision rules derived from the tree are listed below and a visualization of their decision boundaries is shown in Figure 6.22.

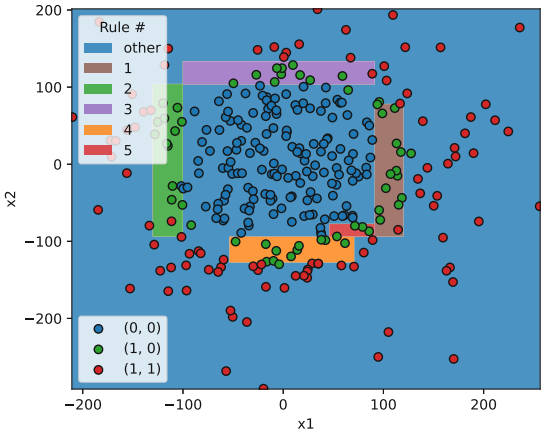


Figure 6.22: Decision boundaries of the 14th baseline explanation, Gaussian Quantiles example. The instances shown are from the explanation data set and are colored by their difference classifier label.

1. if $91.4 < x_1 \leq 120.1 \wedge -94.0 < x_2 \leq 78.1$, then $(1, 0)$
2. if $-130.0 < x_1 \leq -100.1 \wedge -94.0 < x_2 \leq 104.0$, then $(1, 0)$
3. if $-100.1 < x_1 \leq 91.4 \wedge 103.4 < x_2 \leq 133.4$, then $(1, 0)$
4. if $-53.5 < x_1 \leq 71.0 \wedge -127.6 < x_2 \leq -94.0$, then $(1, 0)$
5. if $45.9 < x_1 \leq 91.4 \wedge -94.0 < x_2 \leq -77.3$, then $(1, 0)$
6. else, both classifiers predict the same label

The five rectangular areas approximate the ring shape quite nicely, as can be seen in the decision boundary plot in Figure 6.22. But on the lower left, there is a gap. The explanation achieves 70% F1 score with 20 constraints.

6.3.2 DiRo2C

In total, generation took approx. 39 minutes. A scatter plot of the generated instances is shown in Figure 6.23. The tree explaining $(1, 0)$ has a depth of 18 and offers 23 intermediate steps after pruning. I interpreted the first eight explanations. The decision rules of the last explanation interpreted are shown below and a visualization of the decision boundaries is shown in Figure 6.24. It does not explain the instances #155, #169, #203, #214 and #252. The 20$^{\text{th}}$ Explanation could explain them, but is not easily interpretable, because it contains 87 constraints. You can see it in Figure 6.25.

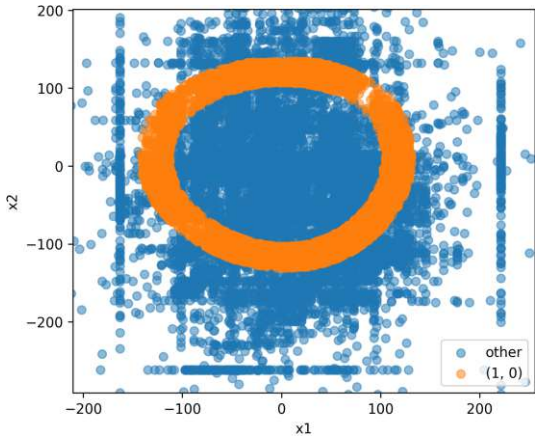


Figure 6.23: Scatter plot of the instances generated by DiRo2C, Gaussian Quantiles example.

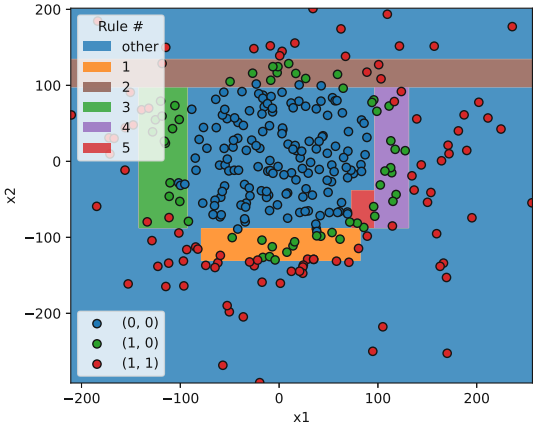


Figure 6.24: Decision boundaries of the DiRo2C 8$^{\text{th}}$ explanation with explanation data set instances.

1. if $-78.9 < x_1 \leq 82.2 \wedge -131.1 < x_2 \leq -87.9$, then $(1, 0)$
2. if $97.5 < x_2 \leq 134.8$, then $(1, 0)$
3. if $-142.0 < x_1 \leq -92.5 \wedge -87.9 < x_2 \leq 97.5$, then $(1, 0)$
4. if $96.6 < x_1 \leq 131.3 \wedge -87.9 < x_2 \leq 97.5$, then $(1, 0)$
5. if $72.7 < x_1 \leq 96.6 \wedge -87.9 < x_2 \leq -37.9$, then $(1, 0)$
6. else, then not $(1, 0)$

The 8$^{\text{th}}$ explanation achieves a F1 score of 70% with 18 constraints. As you can see in the decision boundary plot in Figure 6.24, the rule describing the upper part of the

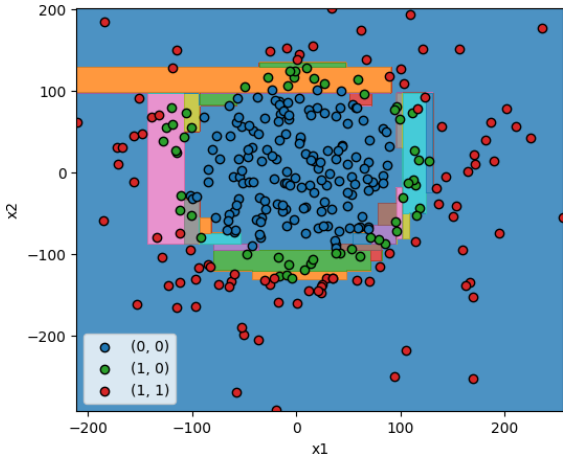


Figure 6.25: Decision boundaries of the DiRo2C 20$^{\text{th}}$ explanation with explanation data set instances.

ring is missing a left and right boundary. In the scatter plot of the generated training instances in Figure 6.23 you can see, that in these two particular areas there are less instances which might have caused the inaccuracy. Also, in the way more complex (87 constraints) 20$^{\text{th}}$ explanation, the error with the missing x_1 boundary on the upper part of the ring still persists on the left side as you can see in Figure 6.25. You can also see, how the decision tree approximates the ring-shaped structure.

6.3.3 Mocca-SHAP

I choose to explain the log of odds output space of the classifiers. In total, generation took approx. 14 minutes. For interpreting the individual classifiers' effects, I just create dependence plots for the effect on class 1's outcome. For interpreting the difference classifier's effects, I select the classes $(0, 0)$, $(1, 0)$ and $(1, 1)$ for brevity and leave out class $(0, 1)$, because no instances are classified like that and no counterfactual explanation suggests that this class could be predicted.

First, I create dependence plots for each individual classifier and for their difference SHAP Values, as shown in Figure 6.26. The effect of both features on the class 1 outcome of both classifiers is very similar. It is shaped concave. But for B, the effect is less steep. This results in B overestimating medium feature values and underestimating low and high feature values, as can be seen in the difference dependence plots in the right column. The instances classified $(1, 0)$ (blue) appear over the entire value range of each feature. From this plot I cannot explain why they are classified differently.

I continue by interpreting the difference classifier dependence plots. The instances classified $(1, 0)$ are split into the cluster nodes RL and RR, and the lowest cluster nodes, that contain the two groups, are nodes RLR and RRRL. The plots for the global level and these two nodes are shown in Figure 6.27.

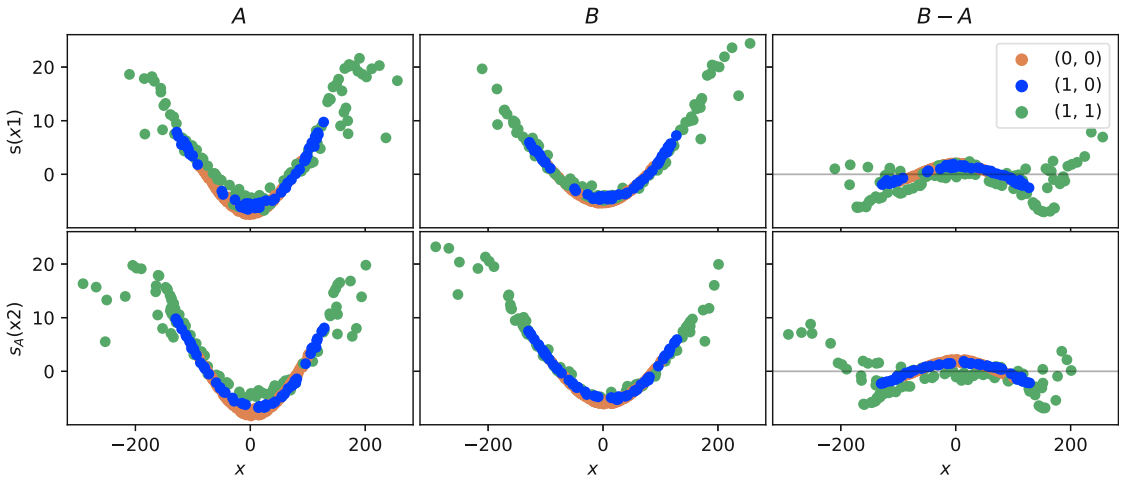


Figure 6.26: Individual and difference dependence plots, Gaussian Quantiles example. The first row shows x_1's effects, the second x_2's effects.

Globally, the two features have a similar shape of their effects. x_1's and x_2's effects on $(0, 0)$'s outcome are shaped convex, while the effects on $(1, 1)$'s outcome are inverted, in a concave shape. The effects on $(1, 0)$'s outcome have two high points: one for lower feature values, and one for higher values. In between, bigger vertical dispersion is present, where the instances classified $(1, 0)$ (blue) have higher SHAP values than other instances, especially those classified $(0, 0)$ (red). The global counterfactual explanations suggest that the differences occur in this specific range between the two high-points of $(1, 0)$'s dependence curves only.

In the the next lower level of the cluster hierarchy, node RLR contains instances classified $(1, 0)$ with either low or high x_1 values and node RRRL those with either low or high x_2 values. Furthermore, node RLR includes mainly instances for which B underestimates the effect of x_1 and node RRRL those, for which B underestimates the effect of x_2. So these two clusters also separate the over- and underestimation effects intertwined in the global difference dependence plots interpreted earlier. The decision rules derived from these cluster nodes are listed below.

- *Root Node*: if $-131.2 < x_2 \leq 136.2 \land -143.8 < x_1 \leq 132.3$, then $(1, 0)$
- *Node RLR*: if $-113.6 < x_2 \leq 123.3 \land -143.8 < x_1 \leq 132.3$, then $(1, 0)$
- *Node RRRL*: if $-131.2 < x_2 \leq 136.2 \land -96.6 < x_1 \leq 93.2$, then $(1, 0)$

This explanation achieves a F1 score of only 39% with eight constraints. This is a lot less than the baseline and DiRo2C. From the dependence plots in Figure 6.27 I can see that there are opposing effects present in the two clusters. Each lower part is increasing monotonically and each upper part decreasing monotonically. I expect each part having

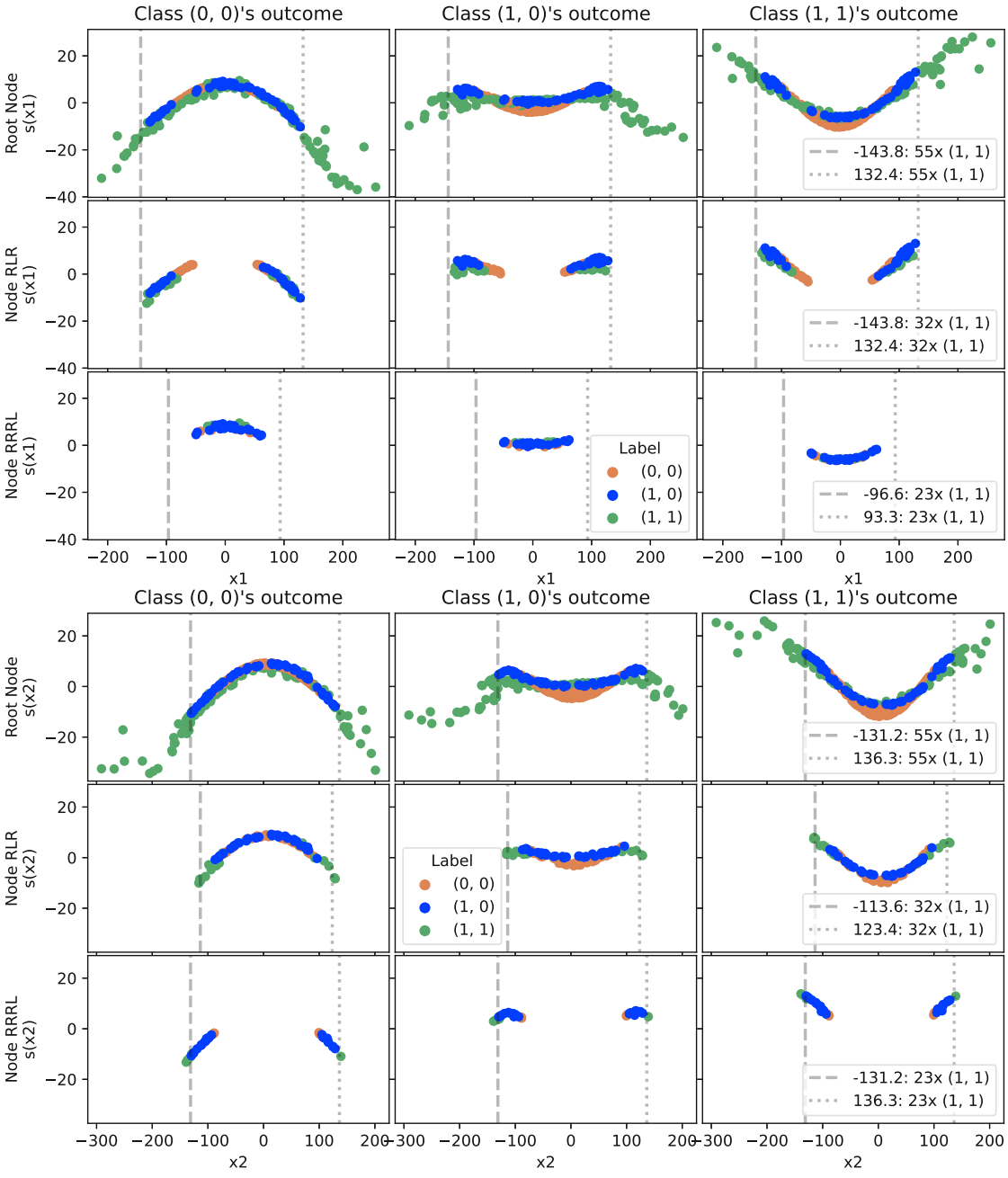


Figure 6.27: Difference classifier dependence plots for the root node, node RLR and RRRL, Gaussian Quantiles example.

different counterfactual explanations, but with the *supervised clustering* approach it is not possible to separate parts like that. With each split, a part of the instances with lower and upper feature values would be put together again, because of their similarity. The solution is to manually split the clusters at the threshold 0. Splitting node RLR by x_1 results in nodes *left* and *right*, and splitting node RRRL by x_2 results in nodes *bottom* and *top*. In Figure 6.28, relevant dependence plots are shown for them. Now the counterfactual explanations are as expected closer to the instances, and also show the boundary to class $(0, 0)$, which was missed before. The decision rules derived from these nodes are listed below.

- *Left*: if $-91.7 < x_2 \leq 108.3 \wedge -143.3 < x_1 \leq -63.2$, then $(1, 0)$
- *Right*: if $-113.6 < x_2 \leq 123.3 \wedge 52.0 < x_1 \leq 132.3$, then $(1, 0)$
- *Top*: if $92.4 < x_2 \leq 136.2 \wedge -96.6 < x_1 \leq 93.2$, then $(1, 0)$
- *Bottom*: if $-131.2 < x_2 \leq -81.2 \wedge -89.4 < x_1 \leq 89.0$, then $(1, 0)$

This explanation now achieves a F1 score of 53% with 16 constraints. This is lower than the baseline and DiRo2C, but is also less complex. In Figure 6.29 you can see in scatter plots of the two features, that node RLR actually contains the instances of the left and right side of the ring and node RRRL those of the bottom and top side of the ring. Figure 6.30 shows scatter plots of nodes *left* and *right* in the first row and nodes *bottom* and *top* in the second row.

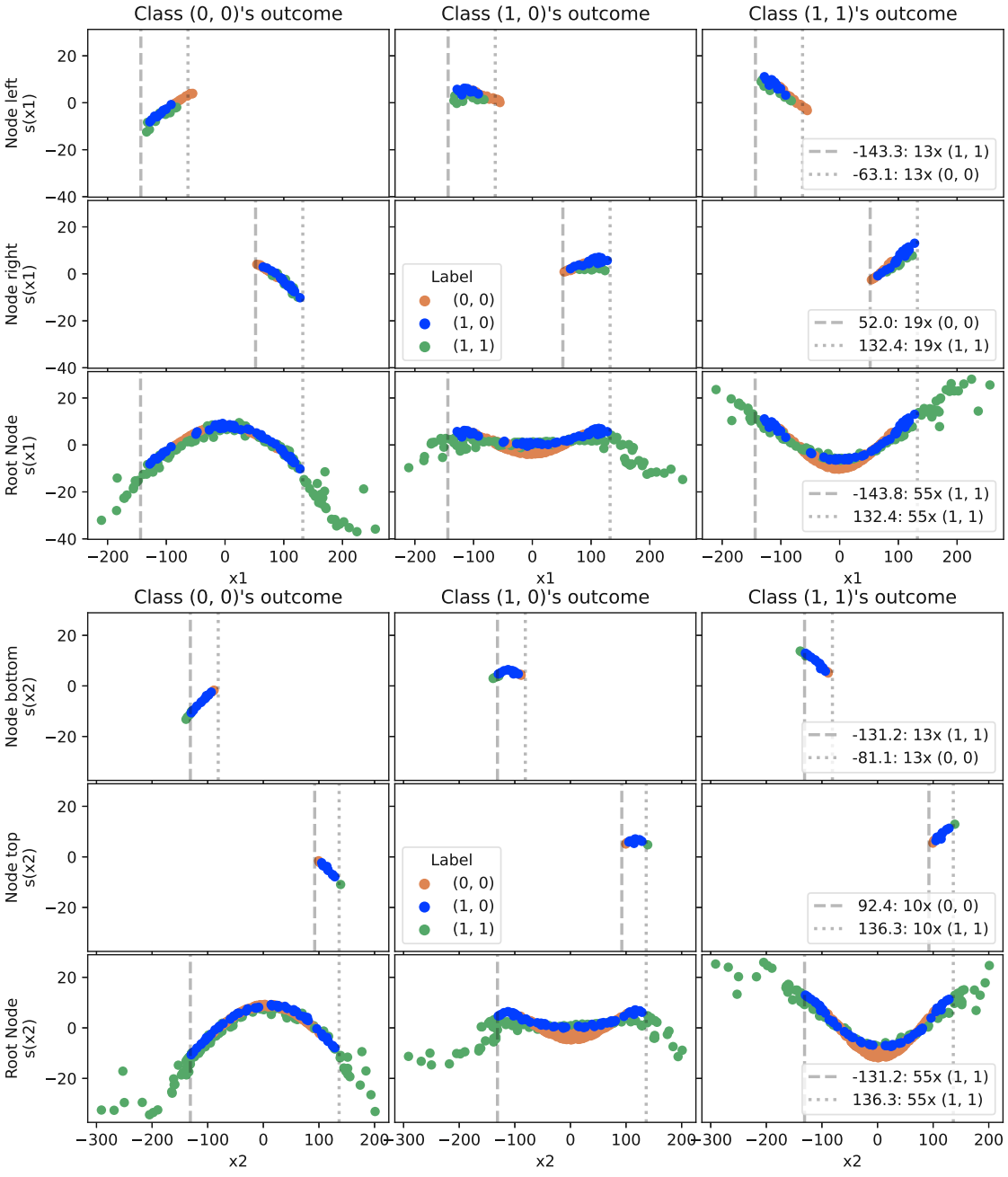


Figure 6.28: Difference classifier dependence plots for the manually interpreted parts, Gaussian Quantiles example.

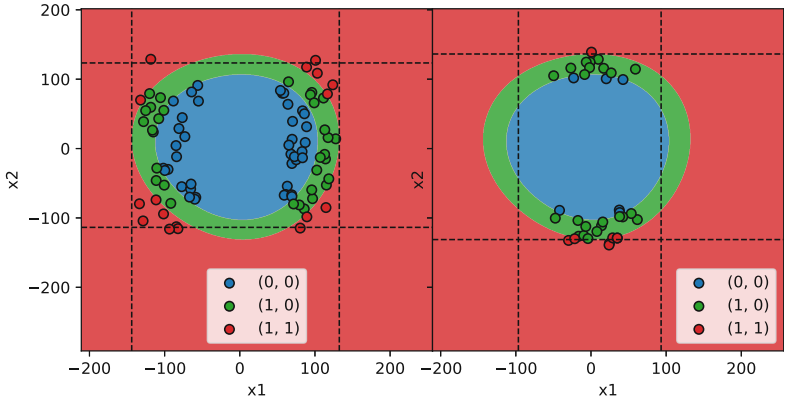


Figure 6.29: Instances of node RLR (left) and node RRRL (right), Gaussian Quantiles example.

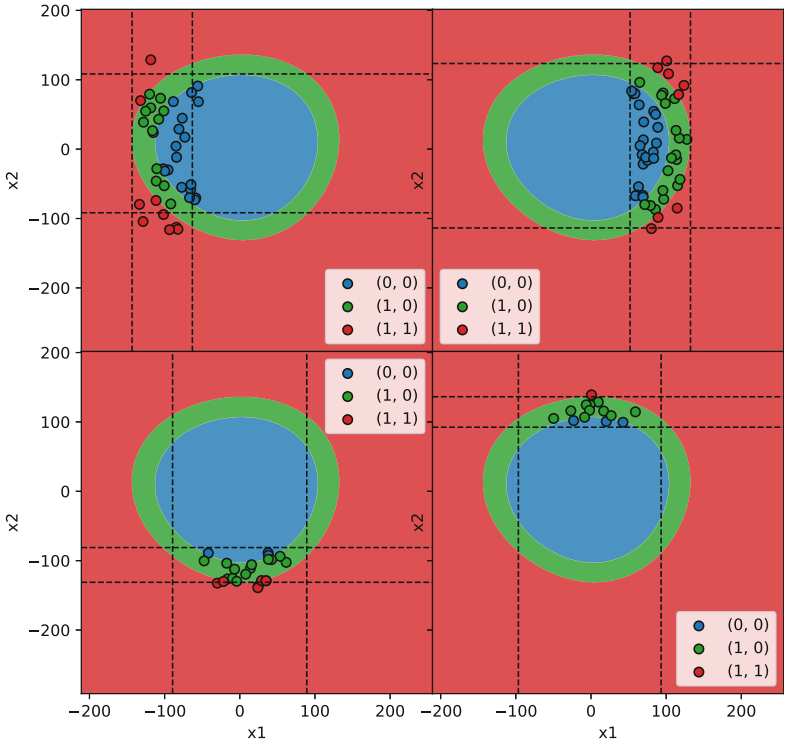


Figure 6.30: Instances of the manually investigated parts, Gaussian Quantiles example. The top row shows the left and right part of node RLR and the bottom row shows the bottom and top part of node RRRL.

6.3.4 Comparison

You can see the results of the automatic evaluation in Figure 6.31. All approaches achieve good results when explaining class $(0, 0)$. As we already know from manual interpretation, Mocca-SHAP is not good at explaining class $(1, 0)$ with low complexity explanations, because of the way the clustering algorithm works. This goes until approx. a complexity of 30, from where it achieves similar performance like the baseline and DiRo2C. The same problem prevents Mocca-SHAP from achieving good fidelity for class $(1, 1)$, where the results are worse. Both the baseline and DiRo2C achieve very good results on this class.

The Mocca-SHAP explanation with the manual workaround achieved a F1 score of 53% with 16 constraints, while the baseline achieved 46% and DiRo2C 41% with 18 constraints on the explanation test set. Therefore, it performs better than the baseline and DiRo2C at this level of complexity.

Generation time of DiRo2C was highest with 39 minutes, followed by Mocca-SHAP with 14 minutes and nearly instant generation of the baseline.

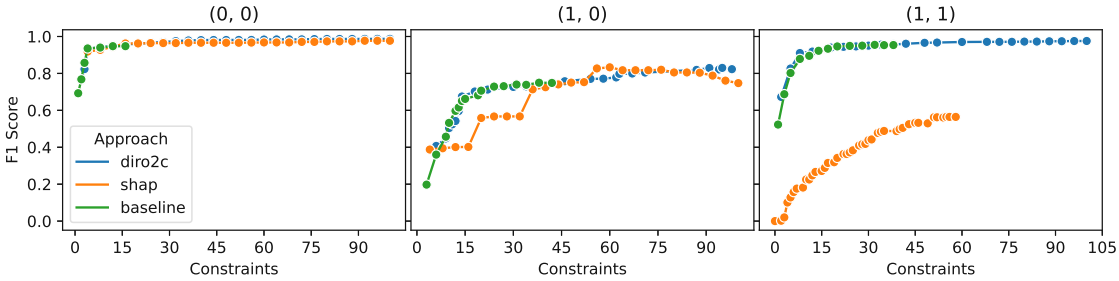


Figure 6.31: Explanation fidelity on the Gaussian Quantiles example test set for explanations with complexities of up to 100 constraints.

6.3.5 Summary

With this example, I have shown how a weakness of the clustering used in Mocca-SHAP can be manually tackled, such that it can achieve a better performance than DiRo2C and the baseline explanations with similar complexity. Also, I have shown that the fidelity of DiRo2C depends heavily on the genetic instance generation algorithm, and that it can worsen the explanation compared to the baseline.

6.4 Census Income (Adult)

In this benchmark task, the goal is to explain why classifier B predicts label *False* more often than A, because its training samples have been manipulated: 10 was added to feature *Hours per week*. Please note, that all thresholds in decision rules are rounded to full numbers for display. These precisions are also used as the step sizes in the Mocca-SHAP algorithm for generating group counterfactual explanations.

6.4.1 Baseline

The baseline approach generated a decision tree with a depth of seven, which, if pruned down to the minimal depth of one, offers ten explanations with different degrees of complexity to interpret. Generation time was less than a second. The three most important features are *Hours per week*, *Relationship* and *Age*, as shown in Figure 6.32. I interpreted all explanations. All instances classified differently are explained by the final explanation. The decision rules derived from the tree are listed below.

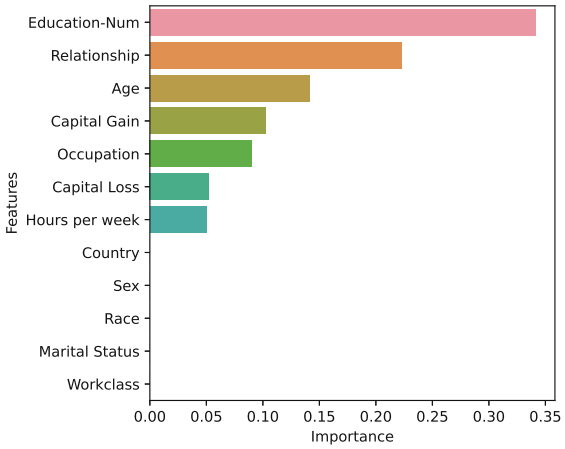


Figure 6.32: Baseline's feature importances of the Adult example.

1. if Education-Num $> 12 \wedge$ Relationship $> 4 \wedge$ Occupation $> 11 \wedge$ Hours per week $\leq$ 48, then $(True, False)$
2. if Education-Num $> 12 \wedge$ Relationship $> 4 \wedge 48 <$ Age $\leq 60 \wedge$ Occupation $\leq$ $11 \wedge$ Hours per week ≤ 48, then $(True, False)$
3. if Education-Num $\leq 12 \wedge$ Capital Gain $\leq 3649 \wedge$ Capital Loss > 2173, then $(True, False)$
4. if $10 <$ Education-Num $\leq 12 \wedge$ Relationship $\leq 4 \wedge$ Age $> 38 \wedge$ Capital Gain $\leq$ $3649 \wedge$ Occupation $> 14 \wedge$ Capital Loss ≤ 2173, then $(True, False)$
5. else, both classifiers predict the same label

To conclude, I can only estimate that certain *Education-Num*, *Relationship* and *Capital Gain* feature value combinations are responsible for the instances classified differently. Actually, these are just the top three most important features for classifier A, as described in the task description in Section 4.4. The explanation does not hint at *Hours per week* as the cause of the differences. It achieves 21% F1 score with 20 constraints on the test set.

6.4.2 DiRo2C

In total, generation took approx. 1.5 hours. Care needs to be taken during interpretation, because DiRo2C has no direct support for categorical or integer numerical features. It treats all features as continuous numbers. This is no issue for the classifiers, because they treat all input features as continuous numbers too. But this leads to unrealistic instances being generated and longer generation time.

A scatter plot of the generated data set to explain class $(True, False)$ and the feature importances are shown in Figure 6.33. The resulting tree has a depth of 16 and offers 88 explanations with different degrees of complexity after pruning. I interpreted the first six. The last of these explains all instances of the explanation data set classified differently except #60. Also, no other explanation with higher complexity is able to explain this instance. The decision rules of the last explanation interpreted are shown below. The last rule can be ignored, because negative *Capital Gain* values are not valid.

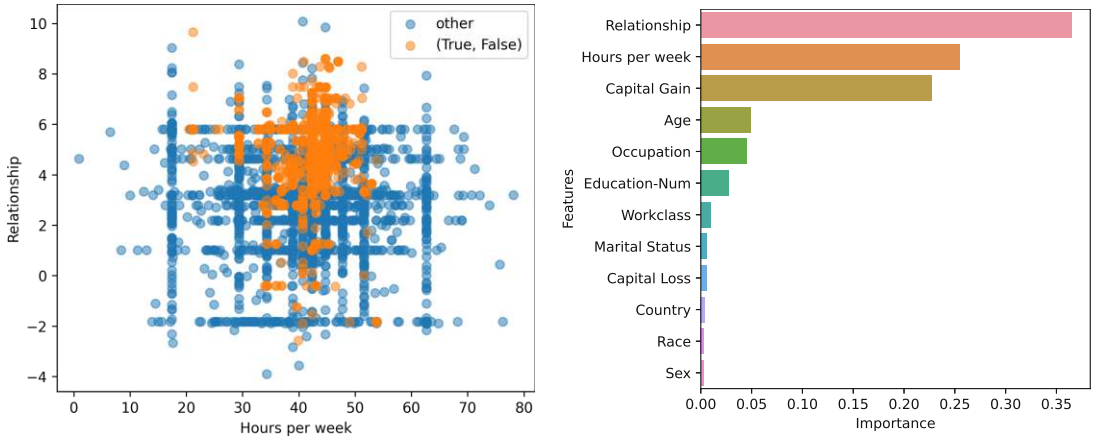


Figure 6.33: Scatter plot of the instances generated by DiRo2C for the two most important features (left) and feature importances (right) of the explanation for difference class $(True, False)$, Adult example.

1. if Relationship $> 3 \wedge 41 <$ Hours per week ≤ 52, then $(True, False)$
2. if Relationship $\leq 3 \wedge$ Capital Gain > 7575, then $(True, False)$
3. if $3 <$ Relationship $\leq 4 \wedge$ Hours per week ≤ 41, then $(True, False)$
4. if Relationship $> 4 \wedge$Capital Gain $\leq -1844 \wedge$Hours per week ≤ 41, then $(True, False)$
5. else, then not $(True, False)$

DiRo2C is able to generate an explanation for class $(False, True)$ too, although no instances are classified like that in the explanation data set. The algorithm only generated

five instances that are classified $(False, True)$ and 13,981 other instances. The decision rules of the second explanation suggest, that this difference class occurs very rarely, only if instances with a certain *Race* have a high *Capital Loss* or instances of certain countries with very high *Hours per week*. The second rule can be ignored, because there is no *Relationship* value of 9.

1. if Race $\leq 3 \wedge$ Capital Loss > 1803, then $(False, True)$
2. if Relationship $= 9 \wedge$ Capital Loss $\leq 1803 \wedge$ Country > 18, then $(False, True)$
3. if Capital Loss $\leq 1803 \wedge$ Hours per week $> 56 \wedge$ Country ≤ 18, then $(False, True)$
4. else, then not $(False, True)$

To conclude, from these explanations I can assume that B estimates label $True$ more often for instances that have *Relationship* equal to `Husband` or `Wife` and work for 42-52 hours, for instances that are neither `Husband` or `Wife` and have a high *Capital Gain* of over 7575 and for instances that are `Husband` and work for 41 hours or less. This interpretation mentions the artificially manipulated feature *Hours per week*, in contrast to the baseline explanation where it seemed only a minor part. It also achieves a marginally higher fidelity with 23% and has eleven constraints (one more than the baseline).

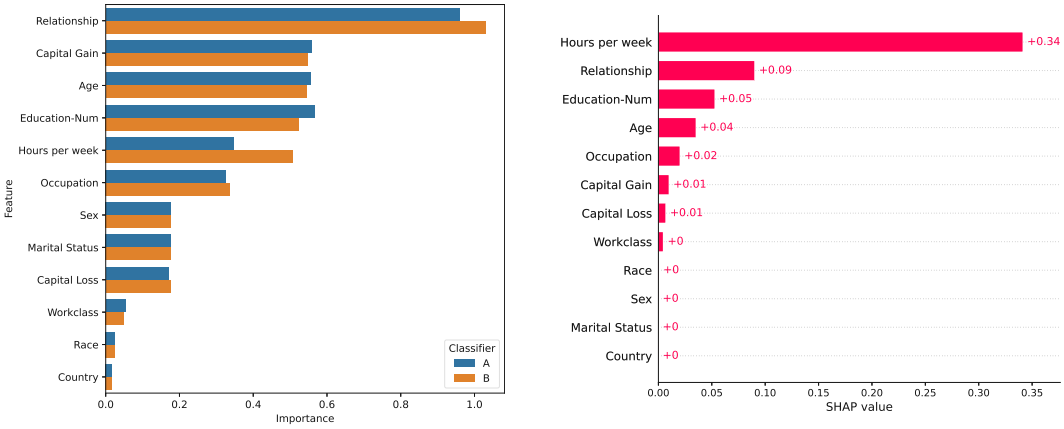
6.4.3 Mocca-SHAP

Generation took approx. three hours. I choose to explain the log of odds output space. First, I plot the feature importances of the individual classifiers, as shown in Figure 6.34a. *Relationship* is the most important feature, followed by *Capital Gain*, *Age*, *Education-Num* and *Hours per week*. *Hours per week* is of higher importance to B than to A and has the highest average absolute differences, as shown in the difference feature importances plot in Figure 6.34b.

I continue by creating the individual and difference dependence plots for feature *Hours per week* (see Figure 6.35), which has the highest difference feature importance. A and B's dependence curves are similar and can be broken down into three parts:

1. Low feature values have a constantly negative effect on the outcome.
2. Medium feature values have a linearly increasing effect on the outcome.
3. High feature values have a constantly positive effect on the outcome.

But B's first part is prolonged in comparison to A's, moving parts (2) and (3) to higher feature values. This results in negative SHAP Value differences around *Hours per week* $=$ 40, which are positive with approx. 0.5 above approx. 50 and below 30. But why is B overestimating the outcome for low and high feature values? This may be to compensate



(a) Individual feature importances.

(b) Difference feature importances.

Figure 6.34: Different feature importance measures, Adult example.

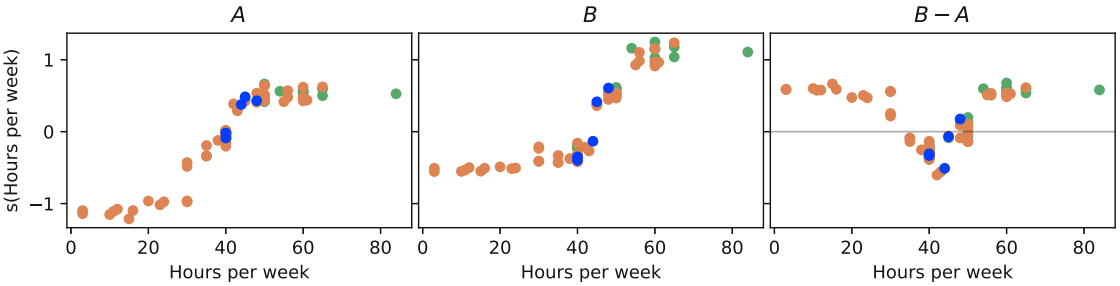


Figure 6.35: Individual and difference dependence plots for feature *Hours per week* and its effect on class $True$'s outcome, Adult example.

for its lower SHAP base value, which is lower by approx. 0.5 than A's. Taking this into account, I can explain why these different classifications occur with just a single feature:

Explanation for the five instances classified $(True, False)$ *(blue):* B tends to estimate label $False$ more often than A because *Hours per week*'s increasing effect on label $True$ happens for higher feature values than in A.

To quantify the offset of B's dependence curve to A's, I move it until both classifiers' dependence curves are aligned above each other, as can be seen in Figure 6.36. This requires to subtract 10 (hours) from the feature values and 0.6 log of odds from the SHAP Values.

To provide a more specific explanation, that takes into account other preconditions that are required for an instance to be classified $(True, False)$, I continue to interpret the difference classifier dependence plots with this class in focus compared to the equality classes $(False, False)$ and $(True, True)$, after generating the clustering structure and computing the counterfactuals. In total, I investigate five cluster nodes besides the root node: RLR, RLL, RLLL and RLLR. Next, I describe my interpretations for the three

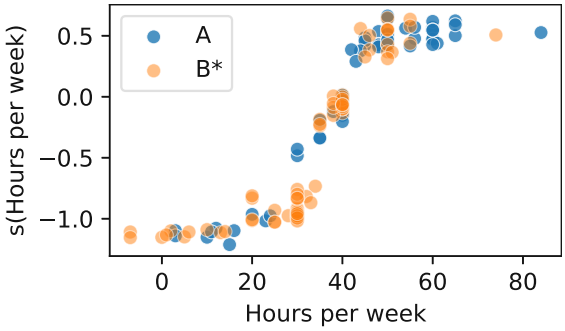


Figure 6.36: Dependence plots of classifier A overlaid by a modified version of classifier B's dependence plot, Adult example.

features with the highest difference feature importances:

Hours per week's global effect on class $(True, False)$'s outcome has a convex shape, while $(False, False)$'s is decreasing and $(True, True)$'s increasing, as can be seen in Figure 6.37. The counterfactual boundaries suggest that the differences are local to the range (34, 55).

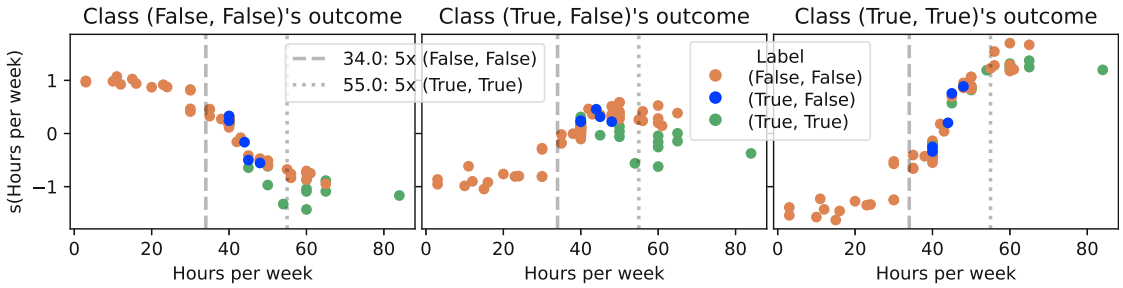


Figure 6.37: Global difference classifier dependence plots for feature *Hours per week*, Adult example.

Relationship is actually a categorical feature, encoded with full numbers. Still, I can analyze the dependence plot, but need to be aware that there is no ordering in the numbers. Its global effect on $(True, False)$'s outcome is decreasing slightly at first until 3, then increasing, as can be seen in Figure 6.38. $(True, True)$'s plot is similar, but steeper and $(False, False)$'s is inverted. Node RLR contains one focus instance with a feature value of 0 and a counterfactual explanation at 3, where the label flips to $(False, False)$ because of the slightly decreasing effect on $(True, False)$'s outcome and the slightly increasing effect on $(False, False)$'s outcome. For a feature value of 4 or 5, the label flips to $(True, True)$ (not shown in figure), because the effect reverses and increases stronger for this class. Node RLL contains the other four focus instances with a *Relationship* value of 4. Their lower counterfactual explanation is at 3, where they flip to $(False, False)$.

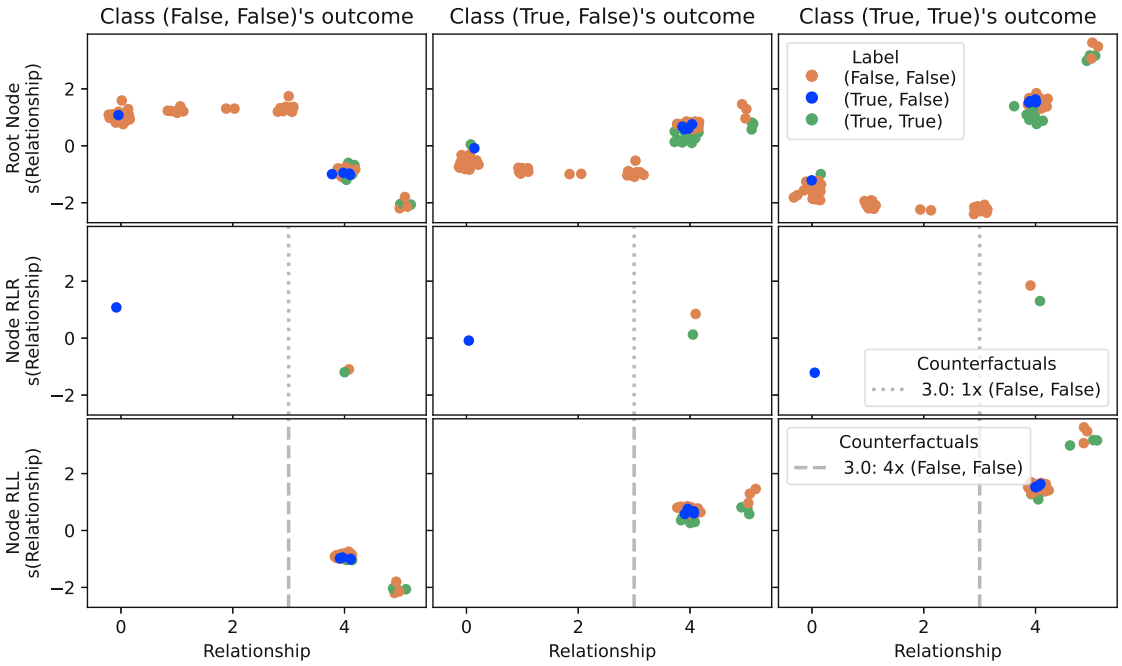


Figure 6.38: Difference classifier dependence plots for feature *Relationship*, Adult example. The first row shows the global effects and the other show local effects.

Education-Num's global effect on $(True, False)$'s outcome is increasing, as can be seen in Figure 6.39. $(True, True)$'s is increasing too, but steeper, while $(False, False)$'s is decreasing. The focus instances have *Education-Num* values between 10 and 13. Nodes RLR and RLLR contain the two focus instances with a lower feature value. At their lower counterfactual boundary, their label flips to $(False, False)$. At its upper counterfactual boundary the focus instance of node RLR flips to $(True, True)$ as expected, but that of node RLLR flips first to $(False, False)$ and then finally to $(True, True)$. As can be seen in the local dependence plots, the curves have local disturbances causing this intermediate label. Node RLLL includes the three focus instances with higher *Education-Num* values. They just have a lower counterfactual explanation at 12, where the label flips to $(False, False)$, as can be seen in the plots of the third row.

Below are the derived decision rules listed, that explain difference class $(True, False)$:

- *Root Node*: if $34 <$ Hours per week $\leq 54 \wedge$ Education-Num $> 8 \wedge$ Age $> 26 \wedge$ Capital Gain ≤ 56, then $(True, False)$
- *Node RLR*: if $34 <$ Hours per week $\leq 44 \wedge$ Relationship $\leq 2 \wedge 8 <$ Education-Num $\leq 12 \wedge$ Age $> 35 \wedge 2 <$ Occupation $\leq 4 \wedge$ Capital Gain $\leq 56 \wedge$ Capital Loss $> 2384 \wedge$ Sex ≤ 0, then $(True, False)$
- *Node RLL*: if $34 <$ Hours per week $\leq 54 \wedge$ Relationship $> 3 \wedge$ Education-Num $> 9 \wedge$ Age $> 26 \wedge$ Capital Gain $\leq 56 \wedge$ Capital Loss ≤ 1447, then $(True, False)$

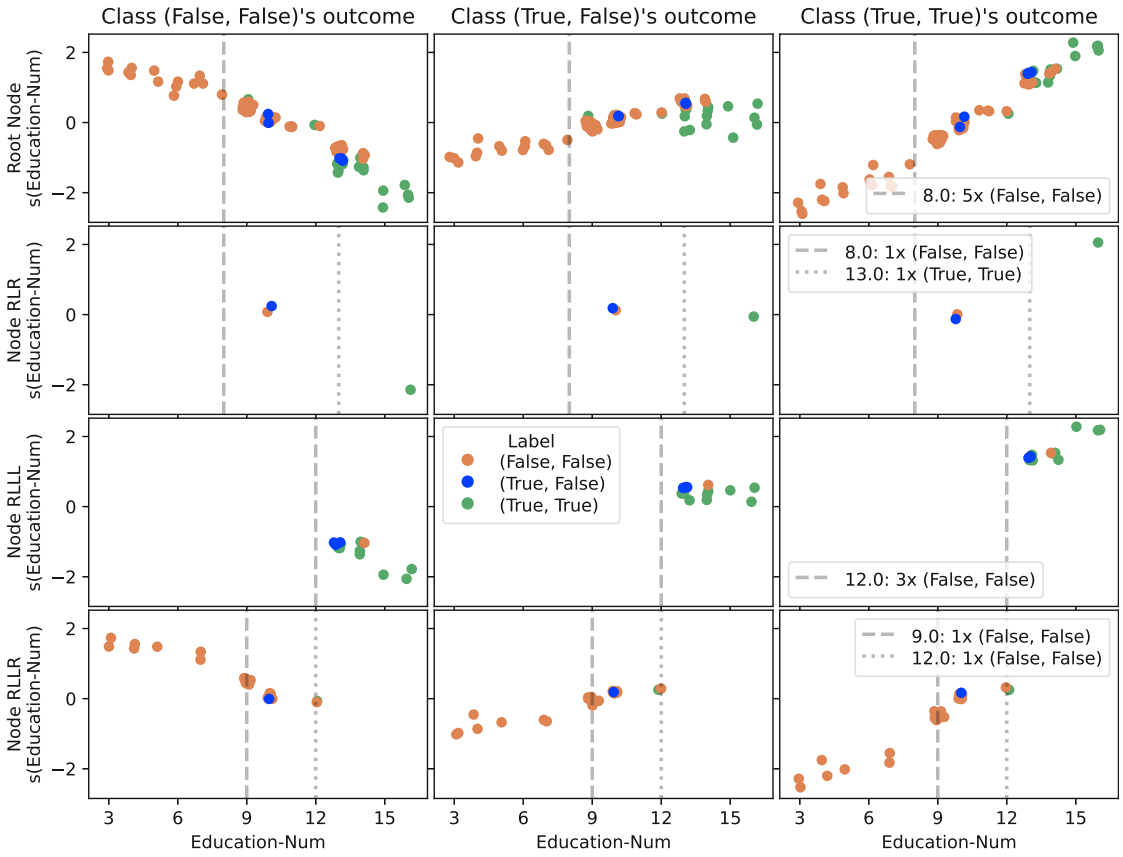


Figure 6.39: Difference classifier dependence plots for feature *Education*, Adult example. The first row shows the global effects and the other show local effects.

- *Node RLLL*: if $34 <$ Hours per week $\leq 45 \wedge 3 <$ Relationship $\leq 4 \wedge$Education-Num $> 12 \wedge$ Age $> 26 \wedge$ Capital Gain $\leq 56 \wedge$ Capital Loss ≤ 1447, then $(True, False)$
- *Node RLLR*: if $44 <$ Hours per week $\leq 54 \wedge$Relationship $> 3 \wedge 9 <$ Education-Num $\leq 11 \wedge 44 <$ Age $\leq 59 \wedge$ Occupation $> 13 \wedge$ Capital Gain $\leq 56 \wedge$ Capital Loss $\leq 1447 \wedge 1 <$ Workclass $\leq 5 \wedge$ Sex $> 0 \wedge$ Marital Status $\leq 2 \wedge$ Country > 0, then $(True, False)$

To conclude, I estimate that the different effect of *Hours per week* is responsible for B classifying more instances as *False* than A, under the condition that it is a value in the range (34, 54], has an *Education-Num* value above 8, an *Age* value above 26 and a *Capital Gain* value of 56 or lower. This explains the artificial modification of *Hours per week* very well. In contrast to DiRo2C and the baseline it also detects that this feature is of highest importance for the differences. I also notice, that none of the approaches explains with just this feature, and that all include other features as well. It might be that it is not possible to attribute the effect completely to the original cause. Mocca-SHAP's

explanation achieves a 54% F1 score with a complexity of 34 constraints, which is much higher than both DiRo2C and the baseline. But it is also more complex, with DiRo2C having only 20 constraints.

6.4.4 Comparison

Generating the Mocca-SHAP explanations took twice as long as the DiRo2C explanations. But most time was spent calculating counterfactual boundaries. With a more efficient algorithm it could be sped up drastically. The fidelities achieved during the automatic evaluation are shown in Figure 6.40.

- Explanations for class $(False, False)$: All approaches achieve a similarly high fidelity for higher complexities. But Mocca-SHAP achieves lower fidelities for explanations with approx. less than 25 constraints.

- Explanations for class $(False, True)$: Although DiRo2C generated explanations for class $(False, True)$, which has no instances in the explanation data set, it achieved a F1 score of 0% for all of them on the test set. But this is an especially hard task, because this is a minority class with only eight instances. The other approaches cannot create explanations for a class without any instances in the explanation data set.

- Explanations for class $(True, False)$: Mocca-SHAP achieves the highest F1 score with 54%.

- Explanations for class $(True, True)$: For low complexities, the baseline and DiRo2C achieve high fidelities (DiRo2C the highest with 71%), but then tend to overfit for explanations with more than approx. 20 constraints and their fidelity decreases again. Here, Mocca-SHAP exceeds with a maximum achieved of 81%.

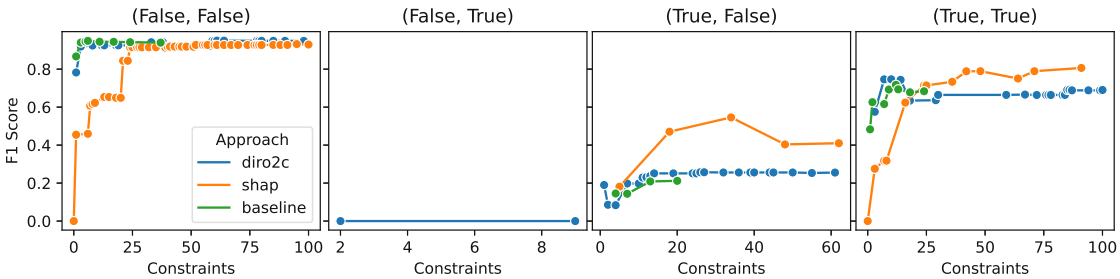


Figure 6.40: Explanation fidelity on the Adult example test set for explanations with complexities of up to 100 constraints.

6.4.5 Summary

Mocca-SHAP not only achieved the highest fidelity on the explanation test set for the explanations interpreted but also the final the highest overall class fidelities. Both DiRo2C and Mocca-SHAP came up with decision rules, that explain that the modified feature *Hours per week* is local to a specific range. Only Mocca-SHAP stated, that this feature is of highest importance for the differences. In the baseline explanation, only half of the rules included one boundary for this feature.

6.5 Boston Housing

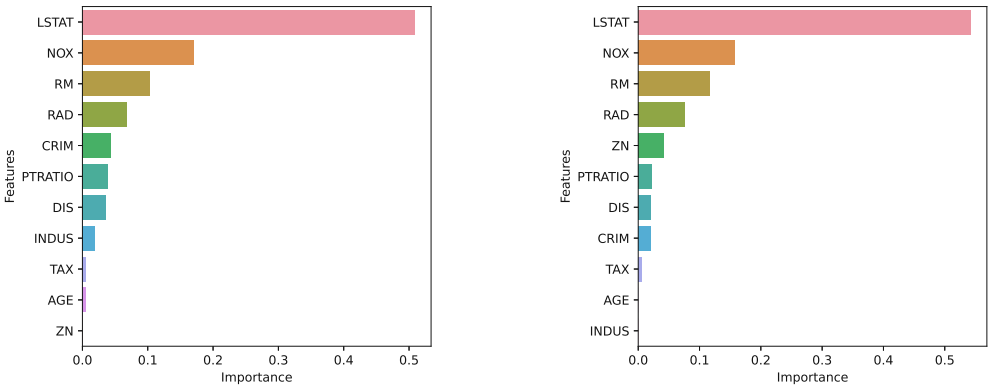
In this final task, I compare the explanations' qualities for multiple multiclass classifiers in a benchmark setting. Classifier A was trained on the original data, B on data where feature *AGE* was inverted and C on data where the label for a cluster selected by LSTAT, ZN and CRIM was changed. There is no test data set for this example, instead I rely on knowledge about the modifications. Please note, that all thresholds in decision rules for ordinal and categorical features are rounded to full numbers, for *CRIM* to one decimal and all other features to two decimals for display. These precisions are also used as the step sizes in the Mocca-SHAP algorithm for generating group counterfactual explanations.

6.5.1 Baseline

Total generation time was less than a second. As can be seen in Figure 6.41, the three most important features in both explanations are *LSTAT*, *NOX* and *RM*.

Comparison of A and B

The explanation for A and B's differences consists of a surrogate decision tree with a depth of eight, which, if pruned down to the minimal depth of 1, offers 19 explanations



(a) A vs. B (b) A vs. C

Figure 6.41: Baseline's feature importances of the Boston Housing example.

with different complexities to interpret. I interpreted all of them, because only the last explanation includes a decision rule that explains the instance classified $(1, 2)$. The decision rules derived from this explanation consist of 41 constraints and are listed below:

1. if $12.61 < \text{LSTAT} \leq 17.72 \land \text{NOX} > 0.59 \land \text{RM} \leq 6.09 \land \text{CRIM} \leq 1.0$, then $(0, 1)$
2. if $9.48 < \text{LSTAT} \leq 12.61 \land \text{NOX} > 0.59 \land \text{INDUS} > 18.84$, then $(0, 1)$
3. if $\text{LSTAT} > 17.52 \land \text{NOX} \leq 0.59 \land \text{CRIM} \leq 1.4 \land \text{PTRATIO} > 20.60$, then $(0, 1)$
4. if $16.72 < \text{LSTAT} \leq 17.52 \land \text{NOX} \leq 0.59 \land 2.27 < \text{DIS} \leq 5.52$, then $(0, 1)$
5. if $\text{LSTAT} \leq 9.48 \land \text{NOX} \leq 0.74 \land \text{RM} \leq 6.62 \land \text{RAD} > 6.00 \land \text{CRIM} \leq 5.0$, then $(2, 1)$
6. if $12.61 < \text{LSTAT} \leq 14.31 \land \text{NOX} > 0.59 \land \text{CRIM} > 1.0 \land \text{DIS} > 2.12$, then $(0, 1)$
7. if $9.48 < \text{LSTAT} \leq 16.72 \land \text{NOX} \leq 0.59 \land 16.20 < \text{PTRATIO} \leq 16.50 \land \text{DIS} > 2.27$, then $(0, 1)$
8. if $\text{LSTAT} \leq 9.48 \land \text{RM} > 6.62 \land \text{PTRATIO} > 20.60$, then $(2, 1)$
9. if $\text{LSTAT} \leq 6.26 \land \text{RM} \leq 6.62 \land \text{RAD} \leq 6.00 \land \text{INDUS} > 4.65$, then $(1, 2)$
10. else, both classifiers predict the same label

Nothing in this explanation hints at the modification done to the training data of classifier B. Instead, it is misleading by assigning high importance to feature *NOX* (ranked second).

Comparison of A and C

The explanation for A and C's differences consists of a decision tree with a depth of eight and offers 17 explanations to interpret. I interpreted the first eleven. The decision rules derived from the last explanation consist of 18 constraints and are listed below:

1. if $16.42 < \text{LSTAT} \leq 16.72 \land \text{NOX} \leq 0.59 \land \text{DIS} > 2.27$, then $(1, 0)$
2. if $\text{LSTAT} \leq 9.62 \land \text{RM} \leq 6.60 \land \text{RAD} \leq 15.00 \land \text{ZN} > 58$, then $(1, 2)$
3. if $9.62 < \text{LSTAT} \leq 11.37 \land \text{NOX} > 0.59$, then $(0, 1)$
4. if $\text{LSTAT} \leq 9.22 \land \text{RM} > 6.60 \land \text{PTRATIO} > 20.60$, then $(2, 1)$
5. if $\text{LSTAT} \leq 9.22 \land \text{RM} > 6.60 \land \text{PTRATIO} \leq 20.60 \land \text{CRIM} \leq 0.0$, then $(1, 2)$
6. else, both classifiers predict the same label

Here too, nothing hints at the modification done to classifier C. Feature *NOX* is again assigned high importance, although it is not involved.

6.5.2 DiRo2C

In total, generation took approx. three hours on the Amazon EC2 instance.

Comparison of A and B

Even though no instances in the explanation data set are classified $(1, 0)$, I also interpret this class' explanation. As can be seen in Figure 6.42, the feature importances of the explanations share one commonality: The two most important features are *LSTAT* and *AGE*. The scatter plots of these two features reveal (see Figure 6.43), that the differences are present in certain areas. Note, that DiRo2C generated unrealistic instances with feature values outside the range of valid values, but this is no issue for the classifiers and has to be taken into account during interpretation.

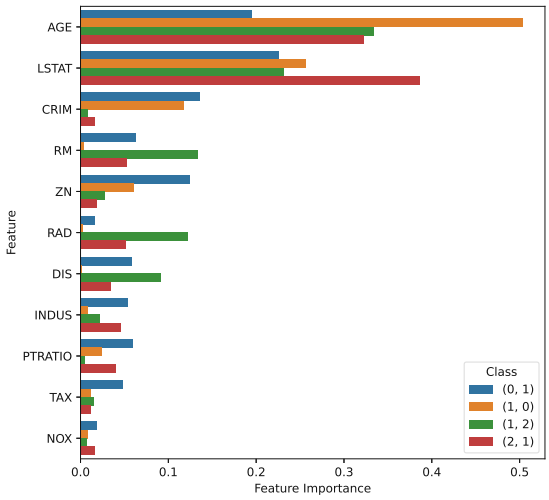


Figure 6.42: A vs. B: DiRo2C's feature importances for the difference class explanations.

Explanation with class $(0, 1)$ *in focus:* The generated focus class instances appear mainly for high *LSTAT* and high *AGE* values, as can be seen in the scatter plot in Figure 6.43. I interpret the first four explanations. The decision rules of the last explanation are listed below. Rules #1 and #2 explain 12 of the 13 instances classified $(0, 1)$ in the explanation data set, but instance #238 is neither explained by this explanation nor by any explanation with higher complexity. But certain constraints are unnecessary: Rule #1 contains a constraint for *CRIM* with a negative threshold, which can be ignored because *CRIM* may assume only positive values. Rule #2 can be ignored, because the highest *AGE* value possible is 2.

1. if $8.53 < \text{LSTAT} \leq 17.18 \wedge \text{AGE} > 1 \wedge \text{CRIM} > -5.7$, then $(0, 1)$
2. if $\text{LSTAT} > 17.18 \wedge \text{AGE} > 2$, then $(0, 1)$

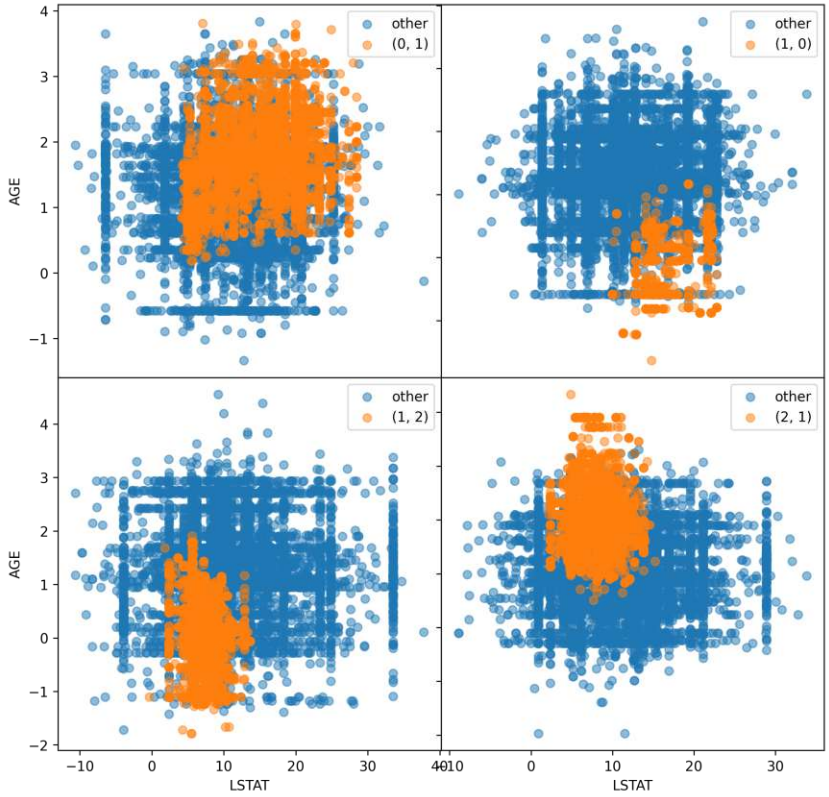


Figure 6.43: A vs. B: DiRo2C's generated instances of the interpreted difference class explanations, *Boston Housing* example.

3. if LSTAT $> 17.18 \wedge 1 <$ AGE ≤ 2, then $(0, 1)$
4. else, then not $(0, 1)$

Explanation with class $(1, 0)$ *in focus:* The generated focus instances appear mainly for high *LSTAT* and low *AGE* values, whereas the explanation data set does not contain any instances classified like that. I interpret the first two explanations, with the last one yielding these decision rules:

1. if AGE $\leq 1 \wedge$ LSTAT > 10.53, then $(1, 0)$
2. else, then not $(1, 0)$

Explanation with class $(1, 2)$ *in focus:* The generated focus instances appear mainly for low *LSTAT* and low *AGE* values. I interpret the first two explanations, with the last one yielding the decision rules listed below. The first rule explains the single instance classified $(1, 2)$ of the explanation data set.

1. if AGE $\leq 1 \wedge$ LSTAT ≤ 11.44, then $(1, 2)$
2. if AGE $> 1 \wedge$ LSTAT $\leq 7.76 \wedge$ RAD ≤ 12.09, then $(1, 2)$
3. else, then not $(1, 2)$

Explanation with class $(2, 1)$ *in focus:* The generated focus instances appear mainly for low *LSTAT* and high *AGE* values. I interpret the first five explanations, with the last one yielding the decision rules listed below. Rule #2 explains two of the three instances of the explanation data set, but instance #51 is not explained by this or any more complex explanation.

1. if $2.38 <$ LSTAT $\leq 12.12 \wedge$ AGE $> 1 \wedge$ PTRATIO ≤ 18.09, then $(2, 1)$
2. if $2.38 <$ LSTAT $\leq 12.12 \wedge$ AGE $> 1 \wedge$ PTRATIO > 18.09, then $(2, 1)$
3. if LSTAT $\leq 9.81 \wedge$ AGE $= 1$, then $(2, 1)$
4. else, then not $(2, 1)$

I conclude that mainly the features *AGE* and *LSTAT* are responsible for the different classifications. B tends to predict a higher label than A for *LSTAT* values greater than 8.53 and *AGE* values of 2, or for *LSTAT* values lower than 11.44 and *AGE* values of 0. B tends to predict a lower label for *LSTAT* values greater than 10.53 and *AGE* values of 0 or 1, or *LSTAT* values in the range (2.38, 12.12] and *AGE* values of 2.

This interpretation lists the actually modified feature *AGE* as being responsible, but without knowing about the nature of the modification.

Comparison of A and C

As can be seen in Figure 6.45, the feature importances of the explanations share one commonality: The two most important features are *LSTAT* and *ZN*. The scatter plots of these two features shown in Figure 6.44 reveal, that the differences are roughly present in separate areas.

Explanation with class $(0, 1)$ *in focus:* Generated focus instances appear mainly for high *LSTAT* and low *ZN* values. I interpret the first two explanations, with the last one yielding the decision rules listed below. The first rule explains two of the three instances of the explanation data set classified $(0, 1)$, but instance #8 is neither explained by this nor by any explanation with higher complexity.

1. if ZN $\leq 23 \wedge$ LSTAT $> 8.62 \wedge$ TAX ≤ 603, then $(0, 1)$
2. else, then not $(0, 1)$

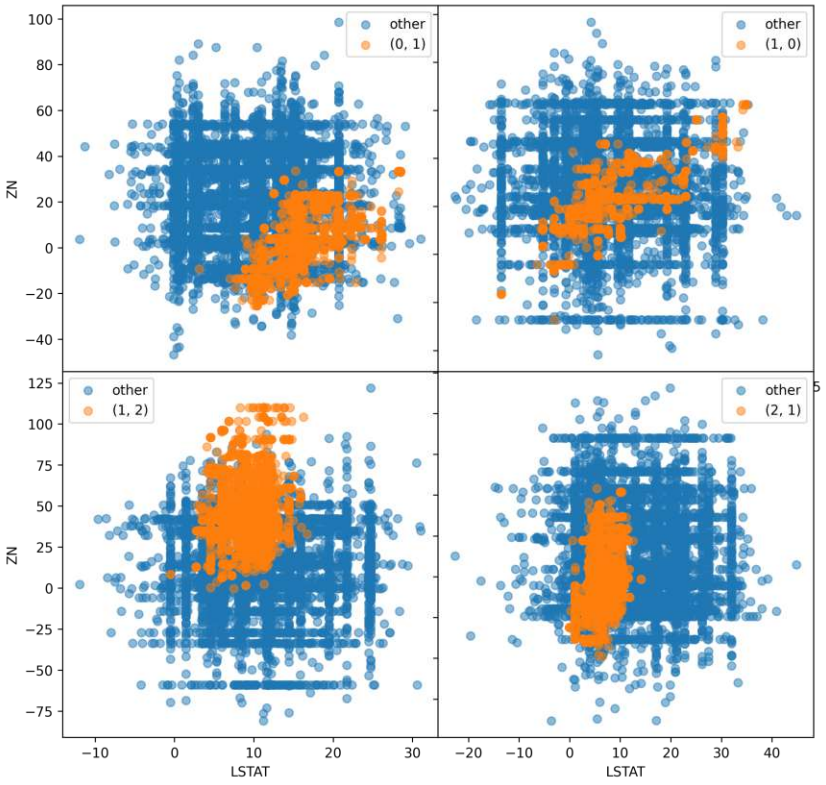


Figure 6.44: A vs. C: DiRo2C's generated instances of the interpreted difference class explanations, *Boston Housing* example.

Explanation with class $(1, 0)$ *in focus:* Generated focus instances appear mainly for medium *LSTAT* and low *ZN* values, and high *LSTAT* and medium *ZN* values. I interpret the first two explanations. Unfortunately, no explanation with higher complexity is able to explain the two instances of the explanation data set classified $(1, 0)$. The decision rules at the last explanation interpreted are listed below. Interestingly, rule #1 is not based on the two overall most important features *LSTAT* and *ZN*.

1. if $\text{CRIM} \leq 7.6 \wedge \text{RM} \leq 6.10 \wedge \text{TAX} > 344$, then $(1, 0)$
2. else, then not $(1, 0)$

Explanation with class $(1, 2)$ *in focus:* Generated focus instances appear mainly for medium *LSTAT* and medium to high *ZN* values. I interpret the first two explanations. The last explanation interpreted consists of the rules listed below, and explains all five instances of the explanation data set classified $(1, 2)$.

1. if $\text{ZN} > 15 \wedge \text{LSTAT} \leq 14.48$, then $(1, 2)$

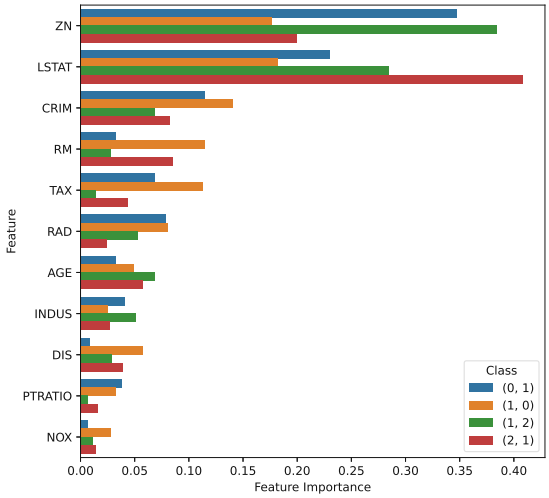


Figure 6.45: A vs. C: DiRo2C's feature importances for the difference class explanations.

2. else, then not $(1, 2)$

Explanation with class $(2, 1)$ *in focus:* Generated focus instances appear mainly for low *LSTAT* and low to medium *ZN* values. I interpret the first two explanaitons, with the last one yielding the rules listed below. The second rule explains the two instances of the explanation data set classified $(2, 1)$.

1. if $3.48 < \text{LSTAT} \leq 10.47 \wedge \text{ZN} \leq 23 \wedge \text{RM} > 6.83$, then $(2, 1)$
2. if $\text{LSTAT} \leq 10.47 \wedge \text{ZN} \leq 23 \wedge \text{RM} \leq 6.83$, then $(2, 1)$
3. else, then not $(2, 1)$

I conclude that mainly the features *ZN* and *LSTAT* are responsible for the different classifications. C tends to predict a higher label than A for *LSTAT* greater than 8.62 and *ZN* lower than 23, and for *LSTAT* lower than 14.48 and *ZN* greater than 15. C tends to predict a lower label than A for low *LSTAT* and low *ZN*; medium *LSTAT* and medium *ZN*; high *LSTAT* and high *ZN*. Furthermore for *LSTAT* lower than 10.47 and *ZN* lower than 23.

This interpretation misses one of responsible features: *CRIM*.

6.5.3 Mocca-SHAP

I choose to explain the log of odds output space. With that, generation takes approx. one hour.

To start, I plot the feature importances of the individual classifiers, as shown in Figure 6.46. They are all similar. But to each classifier, feature *LSTAT* is far more important than any other feature. The individual dependence plots of *LSTAT* (see Figure 6.47) reveal a similar effect across all classifiers. Low values (approx. below 10) increase the chances for label 2. In the range of 5 to 15, label 1's chances are at a peak. And for values above 10, label 0's chances are highest. You can see, that the ranges are overlapping, so other feature values also contribute to a lesser extent. But this feature roughly allows that either labels 0 or 1 are predicted in the lower range up to 10 and that either labels 1 or 2 are predicted above 10.

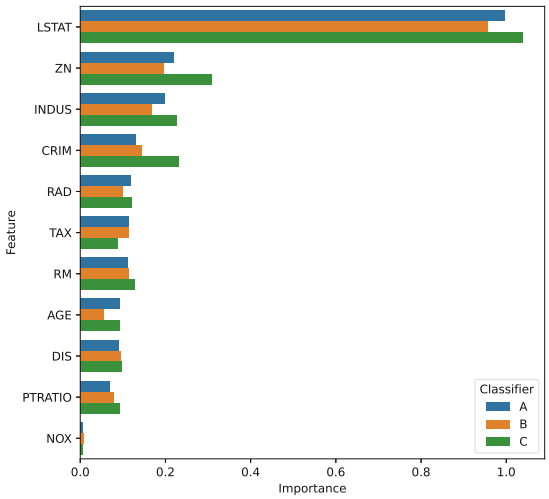


Figure 6.46: SHAP feature importances of the individual classifiers, Boston Housing example.

Comparison of A and B

The main differences between A and B's global influences can be narrowed down to just two features: *AGE* and *LSTAT*. Both have much higher difference SHAP Values than the other features (see Figure 6.48a). It is especially interesting, that *AGE* ranks higher than *LSTAT*, although having only medium influence to each individual classifier.

AGE's individual dependence plots are shown in Figure 6.49. In contrast to *LSTAT*, these are not similar between A and B. For A, $AGE = 0$ increases class 0 and 1's outcomes and decreases 2's. $AGE = 2$ increases class 2's outcome and decreases the others. In contrast, B is estimating an increase in class 1's outcome for $AGE = 2$ and a decrease in its outcome for $AGE = 0$. The other classes' effects are not that clear. This explains, why B is overestimating the label in certain cases:

Explanation for class $(1, 2)$*:* B overestimates the increasing effect of low AGE $(= 0)$ on class 2's outcome. This explains one instance of the explanation data set being classified $(1, 2)$.

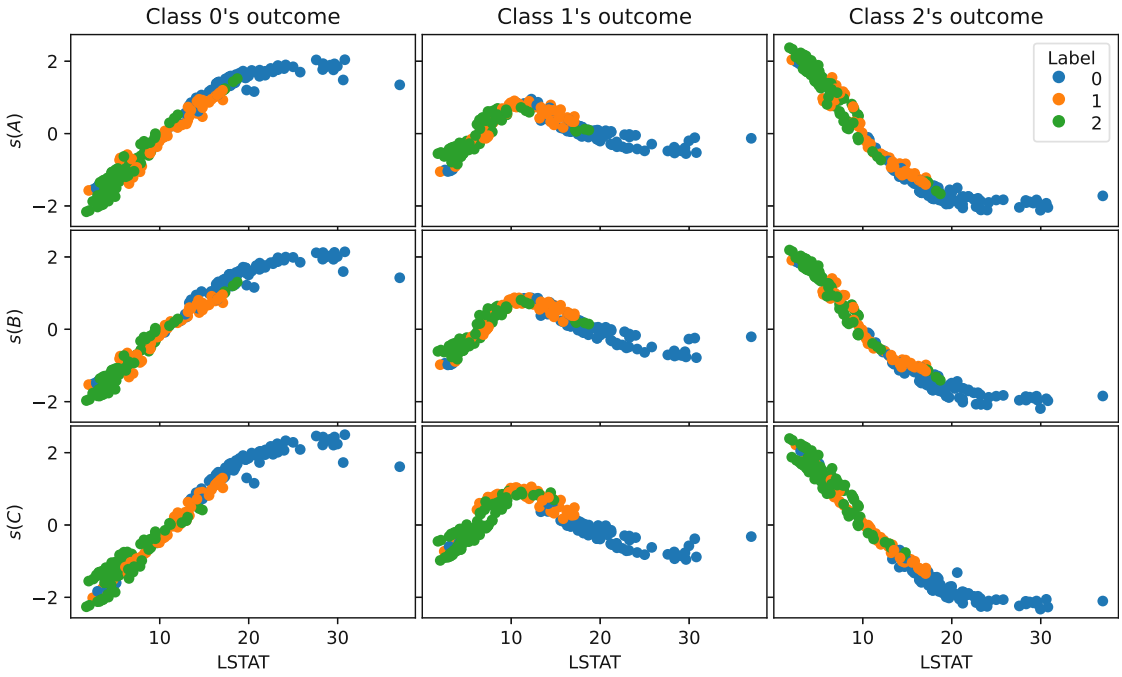
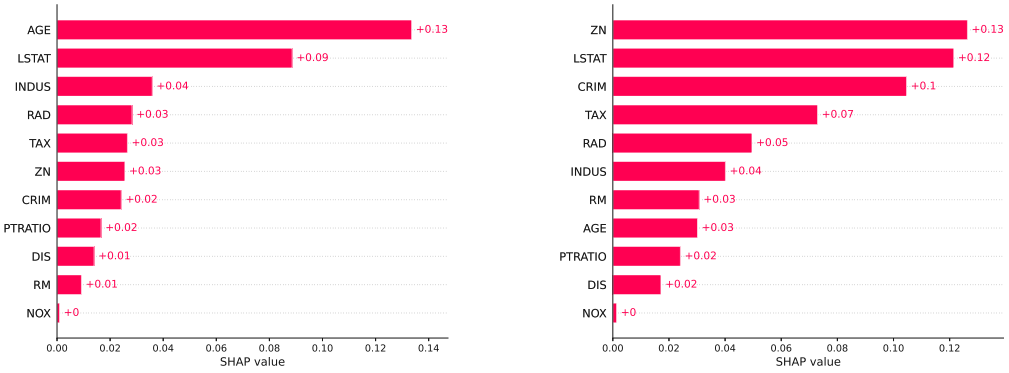


Figure 6.47: SHAP dependence plots of feature *LSTAT*, Boston Housing example. Instances are colored by the label predicted by each classifier.



(a) Comparison of A and B

(b) Comparison of A and C

Figure 6.48: Difference feature importances, Boston Housing example.

The *AGE* effects can be broken down into simpler effects. The first two child nodes of the root node obtained with the *supervised clustering* approach are a sufficient simplification, and can be roughly described by their LSTAT values. Node L contains instances mainly with *LSTAT* values up to 10 and node R all above. As can be seen in the difference dependence plots for these nodes in Figure 6.50, the *AGE* effect on class 0's outcome in node L is decreasing linearly while increasing linearly in node R. This explains, why B is overestimating the label in certain cases:

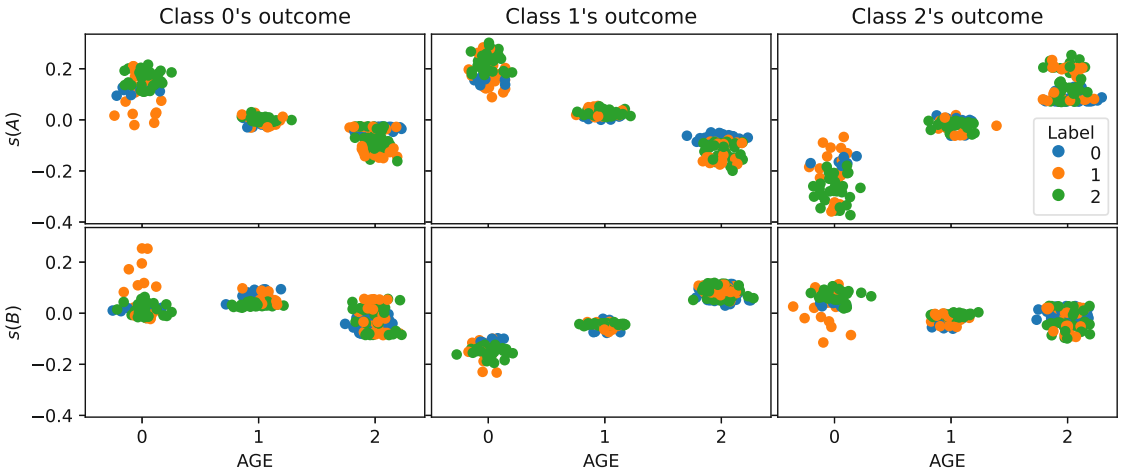


Figure 6.49: SHAP dependence plots of *AGE*, Boston Housing example.

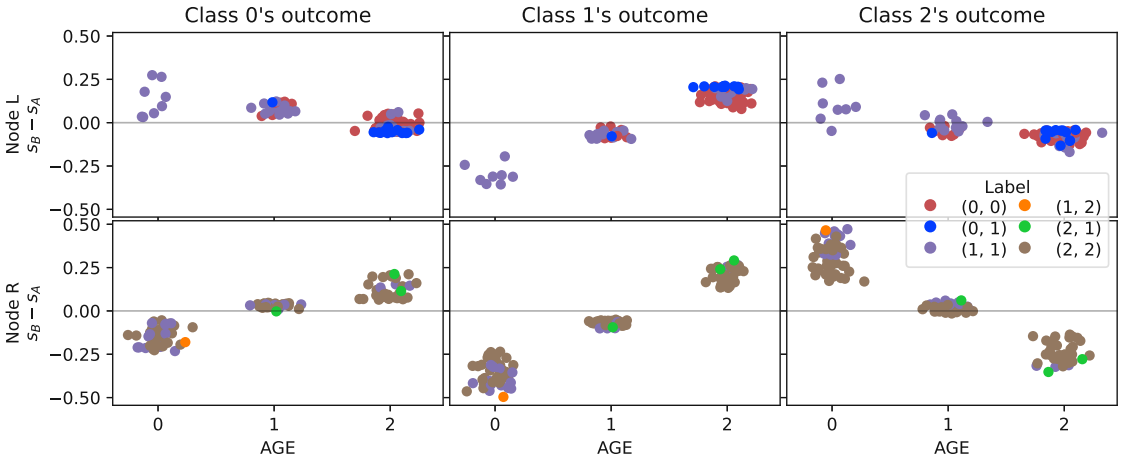


Figure 6.50: SHAP difference dependence plots of *AGE*, Boston Housing example. Each row shows instances of one cluster.

Explanation for class $(0, 1)$: For *LSTAT* values above 10, B underestimates the effect of high *AGE* $(= 2)$ on class 0's outcome and overestimates that on class 1's outcome, which leads to $(0, 1)$ classifications. This explains 12 of the 13 instances of the explanation data set classified $(0, 1)$, which have $AGE = 2$.

Next, I interpret the difference dependence plots of *LSTAT* for node R, which contains instances having *LSTAT* values below 10. The effects can be broken down by plotting separately for each possible value of *AGE*, as shown in Figure 6.51. This reveals, that B overestimates class 2's outcome while underestimating class 1's outcome for $AGE = 0$ in certain cases, and vice versa for *AGE* values of 1 and 2:

- *Explanation for class* $(1, 2)$: For *LSTAT* values below 10 and low *AGE* $(= 0)$, B

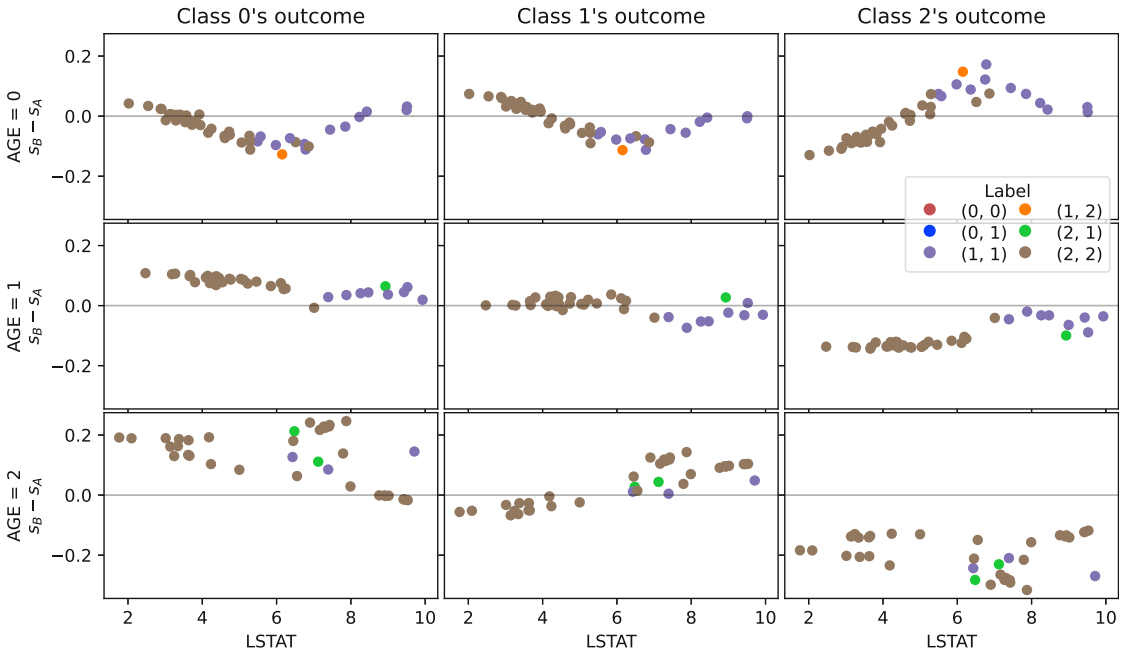


Figure 6.51: SHAP difference dependence plots of *AGE*, Boston Housing example. Each row shows instances of one cluster.

overestimates the effect of *LSTAT* on class 2's outcome, while underestimating the effect on the other classes, which leads to difference class $(1, 2)$. This explains the instance classified $(1, 2)$ of the explanation data set.

- *Explanation for class* $(2, 1)$: For $LSTAT < 10$ and *AGE* values of 1 or 2, B overestimates class 1's outcome and underestimates class 2's outcome. The effect for $AGE = 1$ is not as pronounced as for $AGE = 2$, but still visible. See how the green dots are located above the zero line in the second row, second column and third column and also in the third row, second column and third column. This explains the three instances classified $(2, 1)$ of the explanation data set.

In the same manner, the difference dependence plots of *LSTAT* for node L are split for the different *AGE* values. See Figure 6.52. The data is sparse for *AGE* values of 0, but this does not hinder interpretation:

Explanation for class $(0, 1)$: For $LSTAT > 10$ and *AGE* values of 1 or 2, B is overestimating class 1's outcome and underestimating class 0's outcome. This happens to a greater extent when *AGE* is 2. This explains all 13 instances classified $(0, 1)$.

In the final step I create dependence plots for the difference classifier. For this task, it is sufficient to interpret for each of the three occurring difference classes the global level and the the lowest node that contains all focus instances. For brevity, the plots are not included here, but the decision rules derived from the group counterfactual explanations

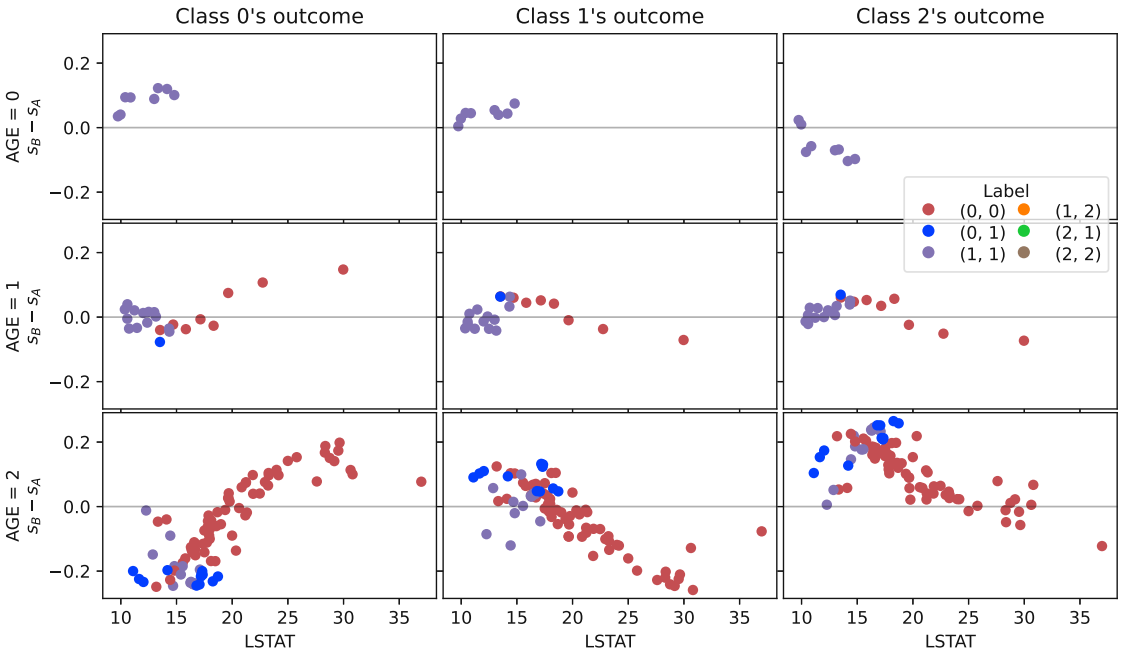


Figure 6.52: SHAP difference dependence plots of *AGE*, Boston Housing example. Each row shows instances of one cluster.

are listed below. Note, that they cannot be interpreted as a list or set of decision rules together, because each describes a difference class in a one-vs-rest manner.

- *Class* $(0, 1)$: if $10.24 < \text{LSTAT} \leq 19.16 \wedge \text{ZN} \leq 5 \wedge \text{CRIM} \leq 6.1 \wedge \text{AGE} > 0$, then $(0, 1)$
- *Class* $(1, 2)$: if $4.51 < \text{LSTAT} \leq 6.27 \wedge \text{ZN} \leq 89 \wedge \text{INDUS} \leq 8.14 \wedge \text{CRIM} \leq 14.6 \wedge 6.20 < \text{RM} \leq 7.74 \wedge 4.27 < \text{RAD} \leq 12.72 \wedge 1.90 < \text{DIS} \leq 5.65 \wedge \text{AGE} \leq 0 \wedge \text{PTRATIO} > 17.34$, then $(1, 2)$
- *Class* $(2, 1)$: if $5.97 < \text{LSTAT} \leq 9.05 \wedge \text{ZN} \leq 81 \wedge \text{INDUS} \leq 19.61 \wedge \text{CRIM} \leq 12.8 \wedge \text{RM} \leq 7.21 \wedge \text{DIS} \leq 4.17 \wedge \text{AGE} > 0$, then $(2, 1)$

I conclude that mainly the features *AGE* and *LSTAT* are responsible for the different classifications. For low *LSTAT* values in the range (4.51, 9.05], B tends to predict a higher label when *AGE* is 0 and it tends to predict a lower label when *AGE* is 1 or 2. For high *LSTAT* values in the range (10.24, 19.16], B tends to predict a higher label when *AGE* is 2 or 1.

This nicely captures the modification introduced into feature *AGE* of classifier B.

Comparison of A and C

Unlike in the previous comparison, there are no features that stand out by their high difference feature importances, as shown in Figure 6.48. Instead, I analyze one after the other, in descending order by their importance, until I can explain all instances classified differently.

The biggest deviations between the individual classifiers' SHAP Values occur in feature *ZN*. This is especially interesting as it is ranked higher than *LSTAT*, which is most important to each individual classifier. Looking at its difference dependence plots (see Figure 6.53), I realize that for values higher than 0, there are big differences in the dependence curves. Since the majority of instances has a value of 0, this concerns only a special part.

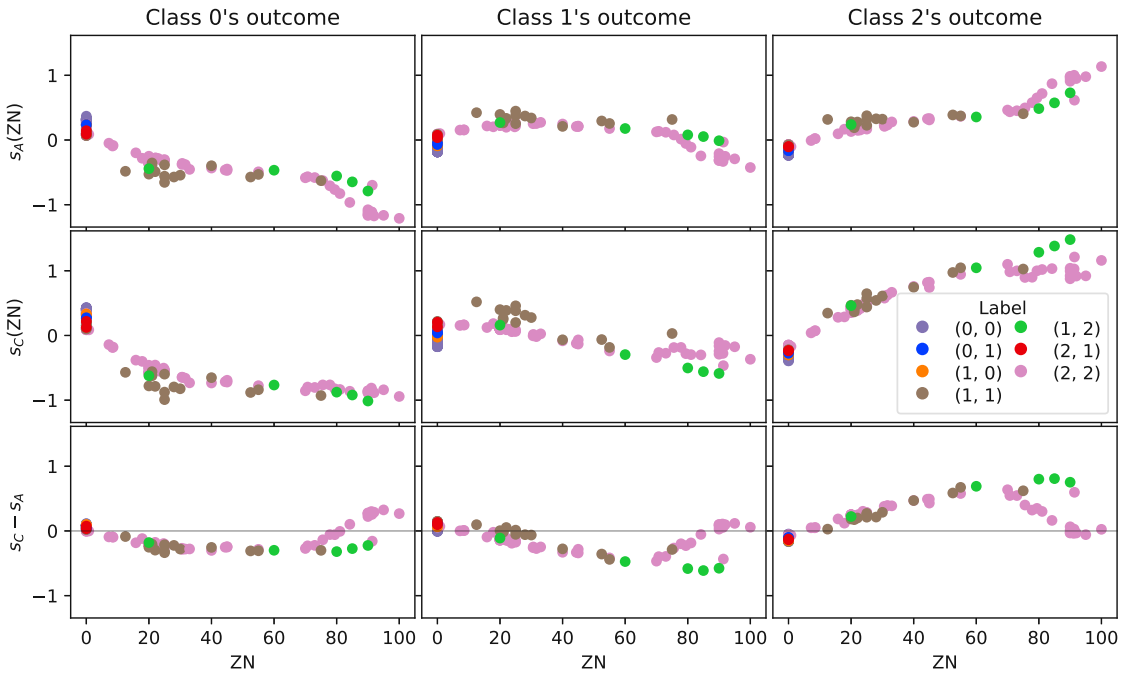


Figure 6.53: SHAP difference dependence plots of *ZN*, Boston Housing example.

Explanation for class $(1, 2)$*:* For $ZN > 0$, C overestimates class 2's outcome and underestimates the others. This explains five instances classified $(1, 2)$ of the explanation data set.

LSTAT's difference dependence plots are simpler and easier to interpret, if I only show instances having $ZN = 0$, as shown in Figure 6.54.

- *Explanation for class* $(2, 1)$*:* Instances classified like that have especially low feature values. At this point in class 2's dependence plot, C underestimates class 2's

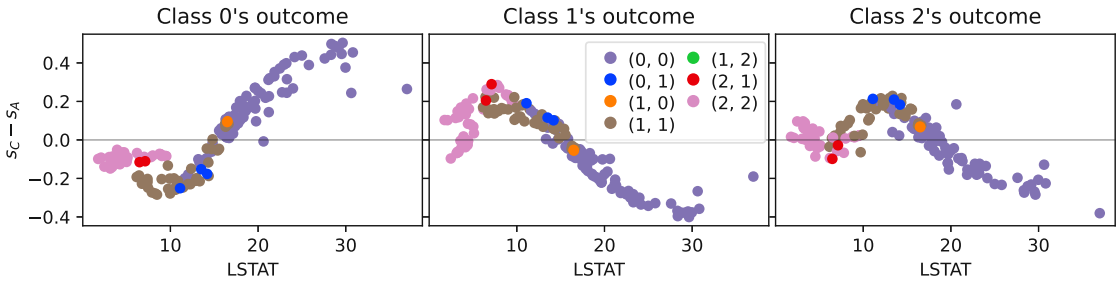


Figure 6.54: SHAP difference dependence plots of *LSTAT* for instances having $ZN = 0$, Boston Housing example.

outcome slightly. At the same time C overestimates class 1's outcome. This explains the two instances classified $(2, 1)$ of the explanation data set.

- *Explanation for* $(0, 1)$*:* Instances classified like that have slightly higher *LSTAT* values. In this range, C overestimates class 1's outcome and underestimates class 0's outcome. This explains the three instances classified $(0, 1)$.
- *Explanation for* $(1, 0)$*:* Instance classified like that again have slightly higher *LSTAT* values, where C switches to underestimating class 1's outcome and overestimating class 0's outcome. This explains the two instances classified $(1, 0)$.

In the final step I create dependence plots for the difference classifier. For this task, it is sufficient to interpret for each of the four occurring difference classes the global level and the the lowest node that contains all focus instances. For brevity, the plots are not included here, but the decision rules derived from the group counterfactual explanations are listed below.

- *Class* $(0, 1)$: if $10.48 < \text{LSTAT} \leq 14.89 \wedge \text{ZN} \leq 1 \wedge 12.37 < \text{INDUS} \leq 22.10 \wedge \text{CRIM} \leq 4.3 \wedge 4.81 < \text{RM} \leq 7.33 \wedge \text{AGE} > 0 \wedge \text{DIS} \leq 6.59 \wedge 13.36 < \text{PTRATIO} \leq 20.73$, then $(0, 1)$
- *Class* $(1, 0)$: if $16.33 < \text{LSTAT} \leq 16.71 \wedge \text{ZN} \leq 0 \wedge 7.13 < \text{INDUS} \leq 9.14 \wedge 0.6 < \text{CRIM} \leq 1.1 \wedge 5.22 < \text{RM} \leq 5.75 \wedge \text{RAD} \leq 5.93 \wedge 288 < \text{TAX} \leq 326 \wedge \text{AGE} > 1 \wedge \text{DIS} > 2.93 \wedge 20.58 < \text{PTRATIO} \leq 21.38$, then $(1, 0)$
- *Class* $(1, 2)$: if $3.58 < \text{LSTAT} \leq 11.58 \wedge \text{ZN} > 16 \wedge \text{INDUS} \leq 16.12 \wedge \text{CRIM} \leq 42.3 \wedge \text{RM} \leq 8.63 \wedge \text{RAD} \leq 13.35 \wedge \text{AGE} \leq 1 \wedge \text{DIS} > 1.94$, then $(1, 2)$
- *Class* $(2, 1)$: if $6.05 < \text{LSTAT} \leq 7.41 \wedge \text{ZN} \leq 18 \wedge 9.80 < \text{INDUS} \leq 19.61 \wedge \text{CRIM} \leq 8.6 \wedge \text{RM} \leq 7.17 \wedge \text{AGE} > 1 \wedge \text{DIS} \leq 4.17$, then $(2, 1)$

I conclude that mainly the features *ZN* and *LSTAT* are responsible for the different classifications. C tends to predict a higher label than A for *LSTAT* values in the range

(6.05, 14.89]. For higher *LSTAT* values in the range (16.33, 16.71], C tends to estimate a lower label. This is true mainly for instances having $ZN = 0$. If the instances have $ZN > 16$, C tends to predict label 2 more often.

Similarly like DiRo2C, Mocca-SHAP identified the features *ZN* and *LSTAT* to be responsible for the differences, but failed to highlight the relevance of *CRIM*.

6.5.4 Summary

Both Mocca-SHAP and DiRo2C provided useful explanations, while the baseline failed to do that. Instead, it is misleading by assigning high importance to feature *NOX*, which is completely disregarded by Mocca-SHAP and DiRo2C.

In the comparison of A and B, both Mocca-SHAP and DiRo2C correctly identified, that *AGE* is of greatest relevance to the differences. Also, both explained in a close relation with feature *LSTAT*, with which it interacts tightly as I found out with Mocca-SHAP. The closest hint at the cause of the different behaviour was shown in the dependence plots in Figures 6.49 and 6.50, because with these I was able to interpret that the classifiers associate a different class order with the *AGE* values. DiRo2C failed to explain the instances #238 and #5, which are boundary cases when considering that they have an *AGE* value of 1. Further note, that only DiRo2C is able to explain difference classes not present in the explanation data set, in this case $(1, 0)$, thus completing the picture.

In the comparison of A and C, both Mocca-SHAP and DiRo2C correctly identified that features *ZN* and *LSTAT* are relevant to the differences, but failed to highlight the significance of *CRIM*, because for both, it was sufficient to just explain using the first two features. Mocca-SHAP ranked *CRIM* third most important in the difference feature importances and DiRo2C fourth most important, so I consider Mocca-SHAP to be a bit closer to identifying it than DiRo2C. Although DiRo2C explains class $(1, 0)$ using *CRIM*, it fails to pick up the relevance of the two other features.

In terms of generation time, Mocca-SHAP was faster by needing only one hour compared to three hours of DiRo2C.

6.6 Summary

With these experiments I answer research question #5.

In the *running* example, I have demonstrated the basic workings of the compared methods. DiRo2C has a weakness, because the fidelity of the explanation is heavily dependent on the initialization parameters in the sampling step. In this particular run, it has scored poorly compared to the other methods, but it may perform very well on another run. Mocca-SHAP beats both the baseline and DiRo2C in terms of maximally achieved fidelity, although its explanation for $(0, 1)$ misses a small part of the feature space. This is because it can only explain differences that are covered with instances in the explanation data set.

In the *One Classifier Ignores a Feature* example, I have shown how the methods perform when confronted with a task, that is hard to describe with step-like explanations. Mocca-SHAP beats the baseline in terms of achieved fidelity on the most detailed level interpreted. I have shown a drawback of it, because its finest level of detail is restricted by the explanation data set, whereas DiRo2C is able to offer more complex explanations with higher fidelity. But both approaches yielded useful explanations. I have also shown, how Mocca-SHAP breaks down the explanation in a modular way into simple local explanations, without the need to interpret single instance's explanations. And I have demonstrated that Mocca-SHAP offers superior interpretation options, because it allowed me to reason about the nature and causes of the differences.

In the *Gaussian Quantiles* example, I have shown how the methods perform on an especially tricky task, where the differences are in the shape of a ring in the two dimensional feature space. Mocca-SHAP makes high errors and mixes the inner circle with the ring heavily because of its *supervised clustering* approach being based on explanation similarity alone. With manual intervention, it achieved higher fidelity than explanations of the other approaches with comparable complexity.

In the *Adult* example, I have evaluated the methods in a benchmark setting with binary classifiers. Mocca-SHAP achieved the highest overall fidelities. Mocca-SHAP and DiRo2C both explained sufficiently by including the feature that was modified, whereas the baseline failed. I have also shown for Mocca-SHAP, how I interpreted in a conversational way each feature's effects. For some features it was sufficient to interpret on the global level, for others I was able to interpret local explanations in the context of the global explanation.

In the *Boston Housing* example, I have evaluated the methods in a benchmark setting by comparing three multiclass classifiers. Both Mocca-SHAP and DiRo2C's explanations have been useful in explaining with the most relevant features, but DiRo2C was not able to explain certain boundary cases, which would have been interesting. DiRo2C hindered interpretation, because it included explanations for unrealistic instances. The baseline explanations were not useful, because they included unimportant features ranked high. I have demonstrated contextuality in a different way here with Mocca-SHAP by exploring different effects for certain feature value combinations, based on a combination of local explanations gathered by the *supervised clustering* approach and manual separation.

Generation time was lower for Mocca-SHAP than for DiRo2C, except in the *Adult* example. A big portion of Mocca-SHAP's generation time is spent searching for group counterfactuals. The algorithm is as of now very inefficient and could be optimized, and in practical scenarios it is not necessary to calculate them for all modular local explanations. I consider SHAP Value generation the limiting factor in the long run. But when switching to faster approximate methods or model-specific methods, this could be sped up as well to be applicable in real-world settings.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

In this master thesis I have proposed the new model comparison method Mocca-SHAP to address the need of data scientists for comparing classifiers with interpretability methods. It is based on the difference classifier, proposed by Staufer and Rauber [Sta21]. I have extended it to support the model-agnostic instance-level interpretability method SHAP. I have shown, that only the multiclass difference classifier variant suits it in a general way. And I have developed an interactive framework with a focus on global-level model understanding with the possibility to investigate local explanations in a modular way, as desired, with the *supervised clustering* approach proposed by Lundberg [LL17]. For this, the instance-level explanations of SHAP Values are visualized together in SHAP Dependence Plots. This framework consists of the traditional explanations for the individual classifiers and explanations for the difference classifier. I furthermore proposed *group counterfactual explanations* to aid in causal interpretations. For evaluation, I compared Mocca-SHAP to DiRo2C [Sta21], which is the original work based on the difference classifier, and a surrogate decision tree that explains the difference classifier as the baseline. For this comparison, I conducted three experiments based on artificial data sets and two experiments based on real-world data sets, all with artificially introduced differences as a ground truth. I measured *Fidelity*, *Complexity* and *Generation Time*. Furthermore, I evaluated qualitatively how Miller's desirable properties for explanations *contrastiveness*, *selectiveness*, *causal links* and *interactivity* are met, which together make up *contextuality*. According to the research question, I show that Mocca-SHAP outperforms the baseline and performs similar to DiRo2C, while offering superior *interactivity* and *contextual* explanations and taking less time to generate in most experiments. I have also shown different strengths and weaknesses of Mocca-SHAP and DiRo2C, which I explain in the following paragraphs in more detail.

DiRo2C is less stable than Mocca-SHAP: In both the *running example* and *Gaussian*

Quantiles example DiRo2C achieved poor fidelity due to bad initialization. But regenerating with different initialization is costly, because of the long generation time.

Supervised clustering is not a general solution: Mocca-SHAP does not perform well on all types of differences. In certain cases, manual intervention was needed, as shown in the *Gaussian Quantiles* example and *Boston Housing* example. This is due to the clustering algorithm used. Potential fixes are discussed in the next section, in 7.2. Because of the conversational, interactive nature of Mocca-SHAP, I do not consider it a big problem.

DiRo2C fails to explain boundary cases: In the experiments, DiRo2C did not come up with explanations for all instances, especially boundary cases were often left unexplained. This did not happen with Mocca-SHAP or the baseline approach.

Mocca-SHAP fails to detect differences not present in data set: It is not able to directly explain differences, that are not present in the data set, while DiRo2C can do that. While both methods are based on instance generation, they may lead to decision rules in DiRo2C but are only taken into account in the feature attributions of the instances to be explained in Mocca-SHAP. This further means, that Mocca-SHAP can only be as accurate as the given data set allows it. I have shown this for DiRo2C in the *running example*, where it worked well. In the *Adult* example, it found explanations, but they did not generalize well.

Contribution compared to the State of the Art

With Mocca-SHAP, I have enriched the research built upon the notion of the difference classifier, which was initially published with DiRo2C [Sta21]. With this I am also contributing to the sparse research that directly seeks to make differences between classifiers interpretable. Furthermore, I have extended the difference classifier for probability estimates.

With this work I have run experiments built on the instance-level interpretability method SHAP and its global extension method SHAP Dependence Plots [LEL19] within new areas of application. I have also investigated operations between SHAP Values of different classifiers with the difference SHAP Values, which are only possible because of the unique properties of SHAP Values that allow them to be treated as explanation models themselves. I have also applied the *supervised clustering* approach by Lundberg [LL17] and built *group counterfactual explanations* on top of their results.

7.2 Future Work

Human-level Evaluation

Because of the time and resource constraints within this master thesis, I ran experiments on my own as a domain expert. Yet, further investigation is needed, to fully test the newly proposed model comparison method. A user study should be done in a follow-up research and evaluate with multiple independent domain experts in the context of specific

real-world tasks with a priori known differences that are to be explained. This can be either with a smaller number of participants in the exact end task or with a larger number of participants in a simplified task [DVK17].

Larger Number of Features

I evaluated only on tasks with data sets, that have a small to medium number of features. But I have shown techniques on how to find features relevant for the differences and proposed multiple importance measures. In a future work, it should be further investigated with data sets including a large number of features.

Comparison of Classifiers of Different Types

In Section 2.1 I have noted, that to avoid additional errors, I am comparing only classifiers of the same type. Yet the method developed treats the classifiers as black boxes, so it is able to compare any classifiers. If all compared classifiers support probability estimates, this output space can be compared as well. But as I noted, they may need to be calibrated to be comparable. Further work should investigate, if fidelity does not worsen too much under these circumstances.

Possible Extensions of the Tools

During development of the tools, I have noted possible extensions and further investigations.

As the clustering algorithm, I have used the same that Lundberg used [LEL19]. But during the *Gaussian Quantiles* experiment, this algorithm failed to break down the explanation space sufficiently, because it is based only on SHAP Value similarity. A different approach could take into account the features and the SHAP Values, which might solve the problem.

In the *Boston Housing* experiment, I have compared more than two classifiers, by breaking down the task into pairwise comparisons. A different approach could extend the difference classification problem to solve this task. It could also be extended for multi-output classifiers.

Because SHAP natively supports image and text data, future work could look into comparison of classifiers with this type of data. Within this limited work, it was not possible to investigate that.

Also, SHAP natively explains regression models. It would be interesting, to see a different problem formulation for difference recognition between regression models, e.g. by explaining the residuals between two models' outputs: $\hat{y}_B - \hat{y}_A$.

As already noted during the experiments, the group counterfactual explanation generation algorithm is in its default implementation very inefficient and could be optimized. And it lacks direct support for categorical feature types.

Finally, I have to note that while the compared methods Mocca-SHAP and DiRo2C are very different in their workings, they have different strengths and weaknesses. Future research could investigate, how they can be combined most efficiently to get a more complete understanding of the differences between classifiers. One option is to apply DiRo2C to the instances of a local Mocca-SHAP explanation. Another option is to use the genetic neighborhood generation algorithm of DiRo2C and calculate SHAP Values for these.

List of Figures

List of Tables

List of Algorithms

Acronyms

ALE Accumulated Local Effects. 9, 49

CRISP-DM CRoss Industry Standard Process for Data Mining. 1, 8

DALEX Descriptive mAchine Learning EXplanations. 21

DiRo2C Difference Recognition of 2 Classifiers. vii, ix, 3–5, 23, 24, 26, 33, 34, 47, 49, 58–60, 63–66, 69–71, 73, 74, 77, 78, 80, 82–84, 86, 89, 91, 96–98, 100, 101, 103, 104, 112, 113, 115, 116, 118, 121–123

GDPR General Data Protection Regulation. 2, 27

KNN K-Nearest Neighbors. 62

LIME Local Interpretable Model-agnostic Explanations. 11, 14, 15

LORE LOcal Rule-based Explanations. 3, 9, 11, 23

MLP Multi-layer Perceptron. 33, 45

Mocca-SHAP Model comparison with clustered difference classifier SHAP values. vii, ix, 3–5, 12, 27, 32–34, 47, 52, 59, 61–63, 65, 68–71, 77, 78, 80, 81, 89, 96–98, 112, 113, 115, 116, 118

PD Partial Dependence. 2, 9, 17–20, 22, 49, 119

RMSE Root Mean Square Error. 1

SHAP SHapley Additive exPlanations. vii, ix, 3, 5, 6, 12, 25, 28, 30, 31, 47–49, 66, 115–117

SOA State-of-the-art. 29, 31, 32

SVC Support Vector Classifier. 33, 40

XAI Explainable Artificial Intelligence. vii, ix, 1, 31

Bibliography

[AB18] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[AR19] Amjad Abu-Rmileh. The multiple faces of 'Feature importance' in XGBoost. https://towardsdatascience.com/be-careful-when-interpreting-your-features-importance-in-xgboost-6e16132588e7, 2019. Accessed: 2022-01-07.

[AZ20] Daniel W Apley and Jingyu Zhu. 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, 2020.

[B+01] Leo Breiman et al. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[BB21] Przemyslaw Biecek and Tomasz Burzykowski. *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. https://pbiecek.github.io/ema/, accessed on 16 November 2021.

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[BC17] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 Workshop on Explainable AI (XAI)*, volume 8, pages 8–13, 2017.

[BFOS84] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification And Regression Trees*. Routledge, 1984.

[Bie18] Przemysław Biecek. DALEX: Explainers for complex predictive models in R. *The Journal of Machine Learning Research*, 19(1):3245–3249, 2018.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

[CG16] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. `http://archive.ics.uci.edu/ml`, accessed on 16 November 2021.

[DMBB20] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

[DSZ16] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617, 2016.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[Eur16] European Commission. General data protection regulation, 2016. `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679`, accessed on 12 November 2021.

[Eur21] European Commission. Artificial intelligence act, 2021. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206`, accessed on 10 August 2022.

[FRD19] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[Fri01] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[GBM18] Brandon M Greenwell, Bradley C Boehmke, and Andrew J McCarthy. A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*, 2018.

[GMR+18a] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[GMR+18b] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5), 2018.

[Hev07] Alan R Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2):4, 2007.

[HGKP20] Patrick Hall, Navdeep Gill, Megan Kurka, and Wen Phan. Machine learning interpretability with H2O driverless AI. *H2O.ai*, 2020. `http://docs.h2o.ai`, accessed on 21 November 2020.

[HJR78] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.

[HMPR04] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, pages 75–105, 2004.

[HTFF09] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

[Hun07] John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[KKK16] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! Criticism for interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.

[KPN16] Josua Krause, Adam Perer, and Kenney Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5686–5697. ACM, 2016.

[LC01] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[LEC+20] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.

[LEL19] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2019.

[LL17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[Lun] Scott M. Lundberg. An introduction to explainable AI with Shapley values. `https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html`. Accessed: 2022-03-14.

[Lun21] Scott M. Lundberg. Exact explainer. `https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/explainers/Exact.html`, 2021. Accessed: 2021-11-22.

[MCB20] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Quantifying model complexity via functional decomposition for better post-hoc interpretability. In Peggy Cellier and Kurt Driessens, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 193–204, Cham, 2020. Springer International Publishing.

[Mil19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Mol20] Christoph Molnar. *Interpretable Machine Learning*. lulu.com, 2020. `https://christophm.github.io/interpretable-ml-book/`, accessed on 21 November 2020.

[Nat07] Shyam Varan Nath. Champion-challenger based predictive model selection. In *Proceedings 2007 IEEE SoutheastCon*, pages 254–254. IEEE, 2007.

[NMC05] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 625–632, New York, NY, USA, 2005. Association for Computing Machinery.

[Pas15] Frank Pasquale. *The black box society*. Harvard University Press, 2015.

[PGG+19] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. Meaningful explanations of black box AI decision systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9780–9784, 2019.

[PHBV08] Jan Pries-Heje, Richard Baskerville, and John R Venable. Strategies for design science research evaluation. In *Proceedings of the European Conference on Information Systems*, pages 255–266, 2008.

[Pla99] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

[Pow08] David Powers. Evaluation: From precision, recall and F-factor to ROC, informedness, markedness & correlation. *Machine Learning Technology*, 2, 2008.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RŠB18] Marko Robnik-Šikonja and Marko Bohanec. Perturbation-based explanations of prediction models. In *Human and Machine Learning*, pages 159–175. Springer, 2018.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[SF20] Kacper Sokol and Peter Flach. Explainability fact sheets: A framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 56–67, 2020.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org, 2017.

[Sha53] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[ŠK14] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014.

[sld] scikit-learn developers. sklearn.tree.DecisionTreeClassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`. Accessed: 2022-03-13.

[SR21a] Andreas Staufer and Andreas Rauber. Gaussian quantiles datasets - DiRo2C. Zenodo, https://doi.org/10.5281/zenodo.5362220, 2021.

[SR21b] Andreas Staufer and Andreas Rauber. Running example datasets - DiRo2C. Zenodo, https://doi.org/10.5281/zenodo.5325335, 2021.

[SSSEA19] Thilo Spinner, Udo Schlegel, Hanna Schäfer, and Mennatallah El-Assady. explAIner: A visual analytics framework for interactive and explainable machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1064–1074, 2019.

[Sta21] Andreas Staufer. *Recognition of Differences between two Binary Black Box Classifiers to create Explanations using Model-Agnostic Methods*. Master's thesis, TU Wien, 2021.

[VDH20] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

[VPHB12] John Venable, Jan Pries-Heje, and Richard Baskerville. A comprehensive framework for evaluation in design science research. In *Processdings of the International Conference on Design Science Research in Information Systems*, pages 423–438. Springer, 2012.

[Was21] Michael L. Waskom. seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[WH00] Rüdiger Wirth and Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39, 2000.

[WMR17] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31:841, 2017.

[YDH19] Fan Yang, Mengnan Du, and Xia Hu. Evaluating explanation without ground truth in interpretable machine learning. *arXiv preprint arXiv:1907.06831*, 2019.

[ZE02] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699, 2002.

[ZWM+18] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):364–373, 2018.