FINAL MASTER PROJECT

Explainability and Fairness in Machine Learning

Presented by:
Fátima del Pilar Villalba Pizarro

Supervised by:
DR. EMILIO CARRIZOSA PRIEGO

FACULTY OF MATHEMATICS
Statistics and Operational Research Department
Seville, June 2021

Contents

Abstract

Over the years, when speaking about prediction models, the focus has been set on improving their accuracy, at the cost of loosing any comprehension of how the model predicts. Consequently, it has also been lost the ability of knowing if the behavior of the model is correct. Moreover, due to the fact that the addresses of the predictions do not have information about how ethic or fair the model is when predicting, persons become reticent to use such type of models. Therefore, in the last years there have been developed investigations aiming to explain such predictions in order to make them intelligible for humans, using techniques like LIME or SHAP, responsible for explaining in an interpretable way what happens behind the prediction. This work addresses this issue and reviews recent literature on the topic.

Resumen

A lo largo de los años, en el ámbito de los modelos de predicción, el foco se ha centrado en mejorar las predicciones realizadas por los modelos, perdiendo a cambio toda comprensión a cerca de cómo el modelo realiza la predicción. La pérdida de comprensión conlleva además el desconocimiento del correcto funcionamiento del modelo, así como reticencias a usar dicho modelo de las personas destinatarias de las predicciones al no poseer información acerca de aspectos éticos y justos a la hora de realizar las predicciones. Es por ello que en los últimos años se ha investigado cómo explicar éstas para así hacerlas de nuevo intelegibles para el ser humano, desarrollando técnicas como LIME y SHAP, encargadas de exponer en una forma interpretable por el ser humano lo que sucede detrás de la predicción. En este trabajo abordamos este tema, y revisamos la literatura existente sobre el mismo.

Introduction

In a world in constant evolution, it is important that predictive models are fast and accurate. Consequently, they have become complex black-boxes, so that their inner working is total opaque for humans. Nevertheless, humans are not just the responsible for using these models to predict, they are also the receptors of such predictions. This is the reason why speed and accuracy are not the only objectives to be considered in for such a model. Behind such a prediction it is necessary to have an explanation supporting the prediction, offering guarantees of correctness and fairness. In this framework the XAI (eXplainable Artificial Intelligence) has been developed in order to explain predictions of black-boxes. Figure 1[1] shows the growing evolution of XAI according to the number of articles that have been published since 2012, including terms related to XAI(*Interpretable Artificial Intelligence*, *XAI* y *Explainable Artificial Intelligence*) in the title, abstract or keywords.

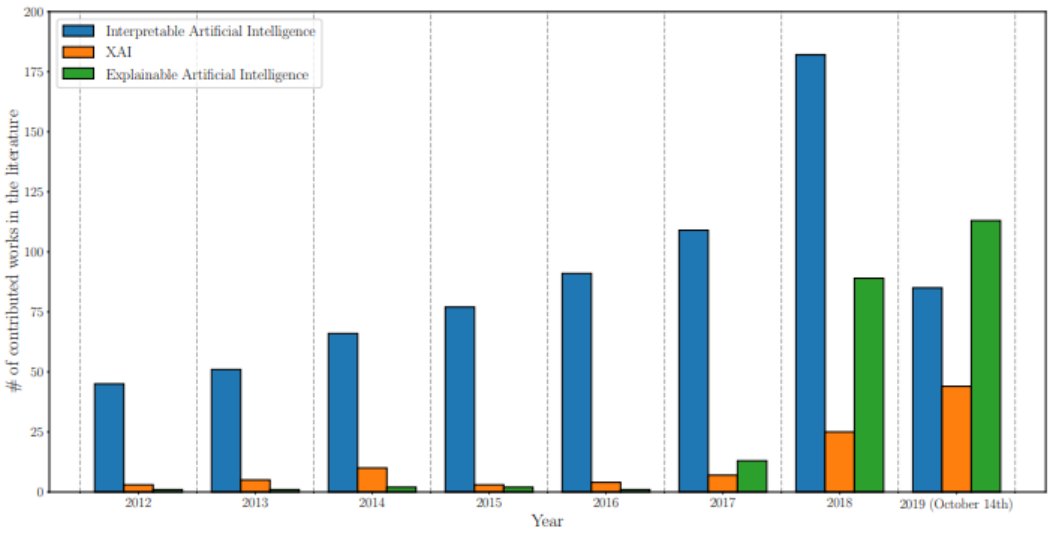


Figure 1: Evolución explicabilidad

Motivated by the need of, not just making accurate predictions, but also explaining

[1]Figure from [5]

them, in this text two relative novel techniques, called LIME and SHAP, will be analyzed. Both are local methods, i.e. methods that explain just a single instance; but in fact, SHAP, as it will be seen in the practical implementation, also gives a global view, explaining not just a single instance, but the whole model.

The text is structured as follows: The Chapter 1 is introduced as an auxiliary chapter, that presents the random forests, as the model that will be used to take the predictions that will be explained afterwards in Chapter 7. Then, in Chapter 2, it is exposed the need and consequently the importance of explanation methods of prediction models. In Chapter 3, the taxonomy of the explanation methods, classifying them into three categories, is exposed. Later on, in Chapter 4, the so-called additive feature attribution methods are introduced as a concrete form of explanation methods. Particular cases of additive feature attribution methods are exposed in Chapter 5 and 6, where LIME and SHAP are developed, respectively. Finally, in Chapter 7, LIME and SHAP are implemented in order to explain the predictions done by random forests for four different data sets, two of which are functional data.

Introducción

En un mundo en constante evolución, es importante tener modelos de predicción que realicen sus predicciones de forma rápida y precisa. Como consecuencia, los modelos son cada vez más complejos, tanto que se han vuelto cajas negras cuyo funcionamiento es totalmente opaco para el ser humano. Sin embargo, el ser humano, además de ser el emisor de los datos que introduce en el modelo, también es, a su vez, el destinatario de dichas predicciones. Esto conlleva que además de rapidez y precisión, se vuelva indispensable que detrás de cada predicción haya una explicación interpretable por el receptor, sobre la que se sustente el por qué de dicha predicción, aportando garantías de que el modelo funciona de forma correcta y justa. En este marco se desarrolla la XAI (inteligencia artificial explicable), que se encarga de explicar las predicciones del modelo y que ha ido tomando relevancia en los últimos años. En la Figura 2[2] podemos ver la evolución desde el año 2012 del número total de publicaciones, cuyo título, resumen y/o palabras clave hacen referencia a términos relacionados con la explicabilidad, en particular para los términos *Interpretable Arificial Intelligence*, *XAI* y *Explainable Artificial Intelligence*, viendo cómo han ido tomando relevancia.

Motivado por esa necesidad de, no solo predecir certeramente, sino entender las bases de la predicción, en este texto se plantean distintas técnicas recientes con este fin, llamadas LIME y SHAP. Ambos métodos son locales, es decir, explican únicamente una instancia en concreto, aunque si bien es cierto, como veremos en la implementación práctica, SHAP proporciona también una visión global, es decir, proporciona una explicación del modelo y no solo de una instancia concreta.

El texto está estructurado de la siguiente forma: El Capítulo 1 es un capítulo auxiliar, en el que se presentan los bosques aleatorios, como modelo de predicción que se usará en la parte práctica (Capítulo 7) para explicar, en base a los distintos métodos que se expondrán en el texto, sus predicciones. Posteriormente, en el Capítulo 2, se plantea la importancia que ha ido tomando la explicabilidad de los modelos de predicción y su motivación. A continuación, en el Capítulo 3, se introducen los distintos métodos de explicación que existen, en base a 3 categorías. En el Capítulo 4, se expondrán los llamados métodos de atribución de características aditivas, entre los que se encuentran distintos métodos de explicación como son LIME, que se desarrollará en el Capítulo 5,

[2]Imagen tomada de [5]

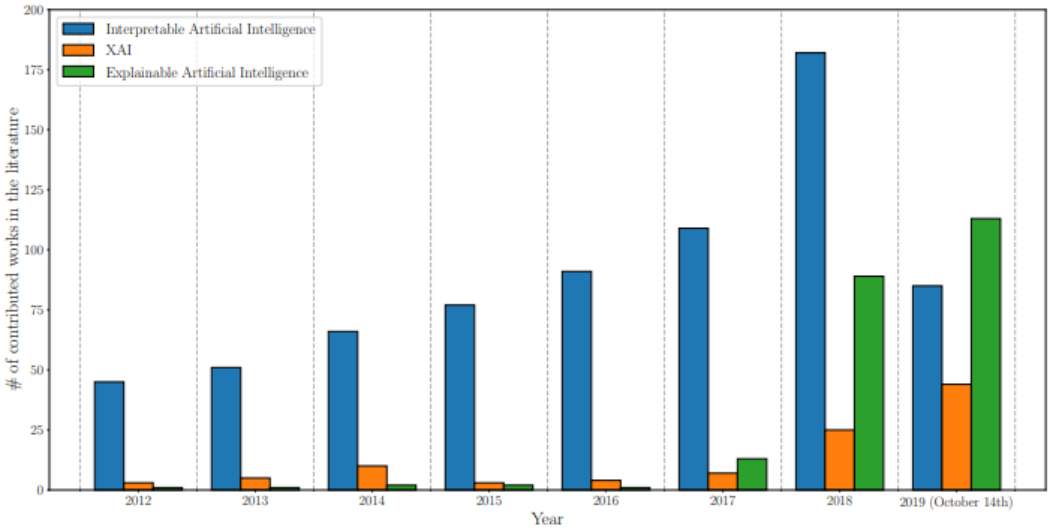


Figura 2: Evolución explicabilidad

y SHAP, que se expondrá en el Capítulo 6. Finalmente, en el Capítulo 7, se implementarán las técnicas LIME y SHAP, para explicar las predicciones de bosques aleatorios para 4 bases de datos distintas, de las cuales 2 son bases de datos funcionales.

Chapter 1

Random forests

In this text different methods for explaining prediction models will be exposed, and in Chapter 7 they will be applied to a specific model, called *random forests*. Therefore let us open a parenthesis and make an overview of random forests.

Random forests yield a compound of two fundamental elements, consisting of an ensemble of decision trees sampled independently (but working together -forest-) where randomness is applied at different points of the algorithm. Let us first understand the way a single decision tree works, and then we will extend the analysis to an ensemble of trees, i.e., to a random forest.

1.1. Roots in CART

The trees used in random forests are grounded on the decision trees presented in 1984 by Breimman and Friedman in "Classification and Regression Trees" (CART) [7]. They predict following an if-then rule, and are therefore easily interpretable [9]. In what follows, we will expose the fundamentals of CART according to [7].

Let $\mathcal{D} = \{(\mathbf{X}^j, Y^j)_{j=1,..,M}\}$ be the learning sample available in order to construct the decision tree, where M is the number of observations, $\mathbf{X}^j \in \mathcal{X}$ is a vector of $|N|$ variables, named measurement vector, in the measurement space $\mathcal{X}$, and Y^j is the corresponding observation: from among R possible classes $\mathcal{C} = \left\{C_1, ..., C_R\right\}$ in case of classification, or $Y^j \in \mathbb{R}$ in regression. Let us see the structure of such a tree and how to construct it.

1.1.1. Structure

The structure of a decision tree consists in a collection of nodes and branches displaying a partition of $\mathcal{D}$, such that the uppermost node is the original set, called

root node t_0, and the other nodes are *terminal nodes*, if they do not split, and *non-terminal nodes* if they do split. Every time a node t_i splits into H nodes (say they are $t_i^h, h = 1, ..., H$) , it is satisfied that $t_i = \sqcup_{h=1}^{H} t_i^h$. We will focus on the trees with $H = 2$, called binary trees.

1.1.2. Construction

1.1.2.1. Splitting criteria

When constructing the tree, the criterion of splitting must be fixed. The most common one for classification trees is the one called *Gini Criterion* and it consists in selecting the split (i.e. as variable X_i for realizing the split) that reduces the impurity at most, understanding as impurity, the probability of classifying wrongly, called by this criterion *Gini Index*. Formally, the Gini Index of a node t is given by:

$$\sum_{r \neq l} p(r \mid t) p(l \mid t) = 1 - \sum_{r} p^2(r \mid t)$$

and the *decrease in impurity* that has to be maximized is:

$$\Delta i(u, t) = -\sum_{r} p^2(r \mid t) + p_{right} \sum_{r} p^2(r \mid t_{right}) + p_{left} \sum_{r} p^2(r \mid t_{left})$$

where $p(r \mid t)$ refers to the probability of assigning an item selected at random from node t to class C_r, and $p(l \mid t)$ expresses the probability of the object belonging to the class C_l, u is the selected split, and p_{right}, p_{left} are the proportion of items of node t that are sent by the split u to the two resulting nodes t_{right} and t_{left}, respectively.

In case of regression the split selected is the one the minimizes the residual sum of squares.

1.1.2.2. Stop-splitting criteria

According to [7], when using individual decision trees, a tree must grow until all terminal nodes are pure, i.e. until only one object has reached one (different) terminal node and then prune upward, i.e., cut off nodes successively in order to select the "optimum-sized" tree. Nevertheless, when constructing a decision tree for a random forest, the tree will not be pruned.

1.1.2.3. Labeling terminal nodes

When doing classification, the terminal nodes are labeled assigning to the node in question the class that share more items of the ones that achieve that node. When doing regression, the node is labeled with the mean of the values of the items.

1.1.3. Random Forests

Once the single decision tree has been explained, it can be set the focus on the whole forest [6], that can bee seen as an improvement when speaking about predictive accuracy, as opposed to decision trees, but at the expense of becoming more complex and less interpretable.

The technique used for constructing a random forest is called *bootstrap-aggregating* and is schemed in Figure 1.1. Given a learning sample $\mathcal{D}$, each tree that will conform the forest is constructed doing a simple random sampling with repositioning. So, for each tree, two new sets are obtained, called *bootstrap* and *OOB (out-of-bag)*. The first one is conformed by approximately $\frac{2}{3}$ of items of $\mathcal{D}$, and the second one, by the $\frac{1}{3}$ of items remaining that do not appear in the bootstrap. The bootstrap is used to construct the decision tree according to what is explained above (Section 1.1), taking into account an additional aspect: the best split is selected, not from the total number of variables available, but from $mtry$ variables selected randomly from among the total. The OOB is used as a validation set of the decision tree, giving the *OOB-error*, i.e. in case of regression, the MSE, and in case of classification, the proportion of wrongly classified items. Afterwards, the results of the different trees are *aggregated*, i.e., on the one hand, the mean of the OOB-errors of all trees is computed, and so the whole forest is validated, and on the other hand, further predictions are given, in regression as the mean of the predictions given by all the trees, and in classification as the most voted class over all the trees.

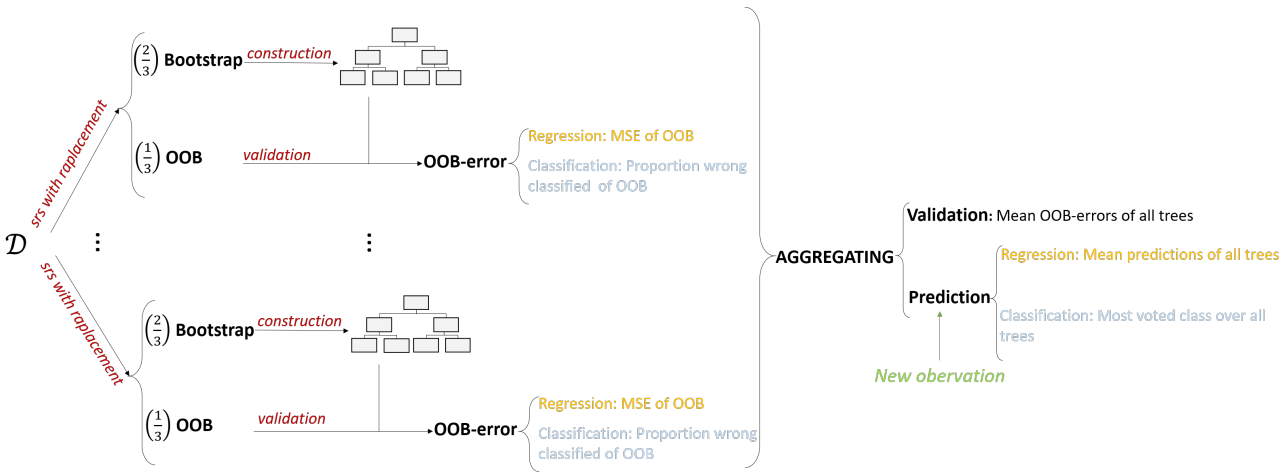


Figure 1.1: scheme random forests

1.1.3.1. Importance of the features

Random forests are an example of prediction models that have evolved from a simple interpretable model with a direct influence of the predictor variables at their position at the tree to a complex one, loosing in interpretability. Consequently, random forests need to be explained. Therefore, addressing the issue of giving an explanation of random forests, there exist three commonly used techniques of measuring the global feature importance:

1. *Split count.* This method counts how many times each features is selected by the trees of the forest as responsible for the split. The higher the number of splits, the more important the variable [36].

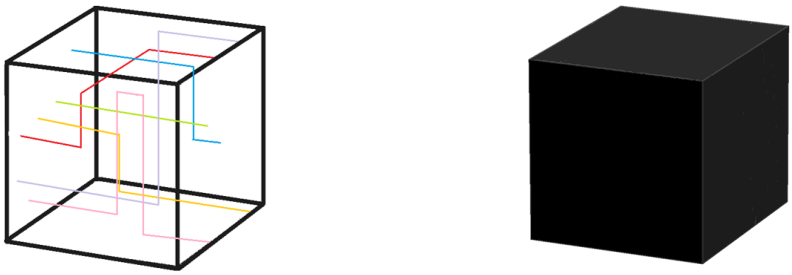
2. *Mean Decrease Impurity (MDI).* This technique averages the decrease in impurity over all the nodes of the trees in the forest, where the variable is the one selected for the splitting. The higher the value of MDI, the more important the variable [29].

3. *Mean Decrease Accuracy (MDA).* This method measures the decrease in accuracy of the forest when permuting the values of the variable whose importance wants to be measured, respect the original random forest. The higher the value of MDA, the more important the variable [17].

Chapter 2

The importance of explainability

Over the years, machine learning has developed and increased it is importance until becoming indispensable in many areas of today's society. This development has been focused on performing the accuracy of the models predictions. Nevertheless, the improvement of prediction accuracy is not cost-free, it is at the expense of making models ever more complex. It is said, that the models have developed from **white-boxes**, to **black-boxes**. An illustration of what is understood under white- and black-boxes is shown in Figure 2.1. While the white-box is transparent, and the structure is clearly visible, the black-box is totally opaque, and the inner structure is not known [26],[27],[5].



(a) white-box (b) black-box

Figure 2.1: Scheme of white- and black-boxes

The main consequence of using black-boxes instead of white ones, is the loss of interpretability, i.e., of comprehensibility, of human understanding of what the "box" is doing to predict [5],[32]. Although it may seem a not important question, since these black-boxes have a high predictive accuracy, and they "just do it", due to the fact that predictive models are used in several areas that have immediate effects on human life, like medicine, finance, security, transportation, etc [5],[32],[33], understanding "what happens behind" is an important task. In order to evince it, let us see the following

example, schemed in Figure 2.2. Imagine a doctor is making the pacient's diagnosis in agreement with a model prediction. If that model is a white-box, the doctor knows how the model is working, and consequently why *that* prediction was made. Nevertheless, probably (not necessarily)[1], the accuracy of the prediction will not be that high. In contrast, if the model is a black-box, the prediction accuracy will increase, but the doctor will not know the explanations supporting the output of the model, so: Could the pacient feel safe in this situation?

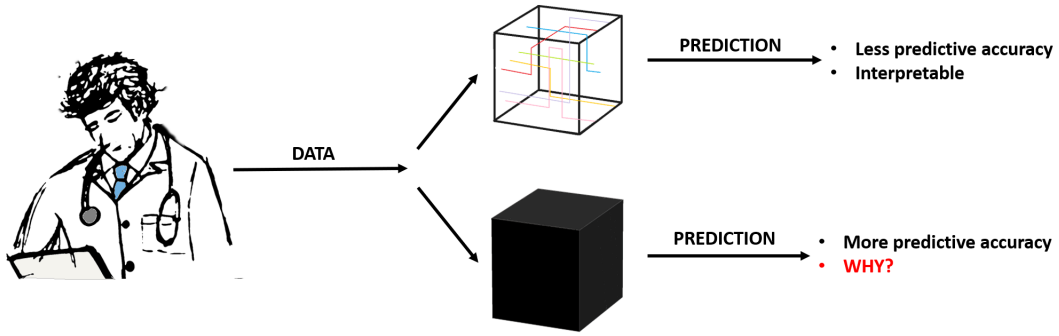


Figure 2.2: Scheme of example doctor

The open-ended question of the above example brings out that humans are reticent to non interpretable procedures [5] whose trustworthiness cannot be verified [2] but have an immediate consequence on their life. The consequence is that if a human cannot trust a model, it will not use it, because, although the lack of trustworthy does not imply a lack of efficiency, efficiency does also not imply that it cannot be faulty [30].
An example is given in [31] where the authors trained a classifier that should classify images, depending on the animal appearing on them, in two classes: wolves and huskies. Nevertheless they did not trained this model with random images, they did that with 20 images such that, at every image where a wolf was appearing, it also appeared snow in the background. So, when realizing further predictions, the model used as principle criterion to distinguish between both classes if there was snow in the background, but any aspect about the animals. The predictions were mainly correct, because as test they also used images where the background of the wolves was also snow. If the focus is set just on the accuracy of the model, it seems to be a good model, but when the explanations of the model are exposed (predicting according to snow), the model is revealed to be a "bad" model, that can fail and should not be trusted.

[1] In [32] it is discussed about *Interpretable Machine Learning* a term coined in the 1950's based on the idea of using inherently interpretable models, and it is highlighted that these type of models are not necessary at odds with high predictive accuracy, focusing on the idea that sometimes black-boxes are implemented unnecessarily.

[2] In [32] it is exposed that interpretable models do not necessary imply trustworthy, but they *allow to check* if they are trustful.

Furthermore, related with the possibility of failure, in many areas there exists a legal aspect that demands "right to explanation", because there also exists the "right of human intervention" [30].

Moreover, a model is expected to be "*fair and ethical*" [30]. Suppose a model is used to predict the best applicant for a job. The prediction should not be taken according to aspects like age, sex or religion, because these would not be ethical, bet neither fair [8]. So if a candidate is refused, it has the right to ask "Why?", and get an explanation, otherwise if there is no explanation of what the model is doing, the model cannot be trusted to be fair and ethical, calling the model into question [30].

Therefore, like black-boxes make high accurate predictions, but fail on interpretability, i.e., on making predictions *justifiable* [5],[30] based on their working , so that they can be confirmed to be trustworthy [5],[27],[30], in around 2016 arises the XAI (explainable artificial intelligence) [32] with the aim of addressing the trade-off between predictive accuracy and interpretability [3] [32],[24],[25],[5],[21], for the different predictive models that have been and will be developed over the years, like the ones of Figure 2.3 [5].

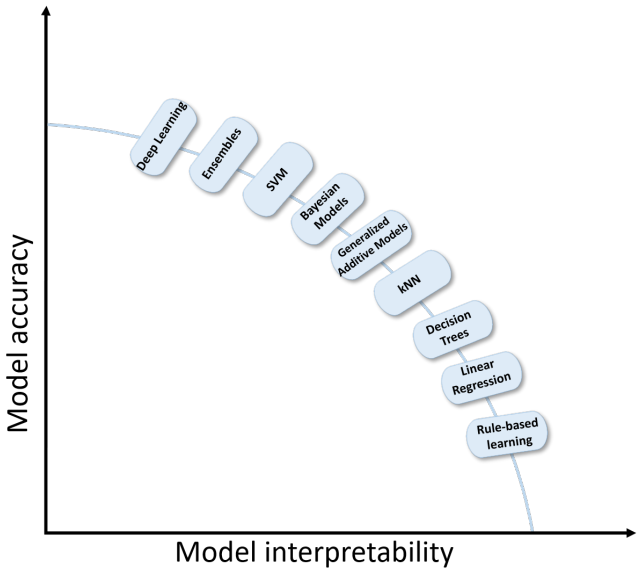


Figure 2.3: Models accuracy vs. interpretability (evolution)

It is worth emphasizing the term appearing in XAI, *explainable*, since it should not be confused with *interpretable*. Interpretable means that the model *is* comprehensible

[3]Some authors call this readability-performance trade-off or accuracy-comprehensibility trade-off instead of accuracy-interpretability trade-off [30]

or understandable for the human (a white-box), while explainable must be assimilated as that it *can be* comprehensible, i.e., interpretable involves the concept of inherently, while explainable implies that the model becomes interpretable *after* a certain analysis, i.e., the explanation is interpretable. So, the explanation implies interpretability, but not the other way around. Even though, both concepts are closely tied to the target audience, and therefore, to take full advantage of prediction models, the model should either be interpretable/comprehensible for the target audience, or the explanations given afterwards should be presented in a understandable way for them. Nevertheless, there does not exist a formal definition of both concepts, and therefore, different authors make different assumptions about the different meanings [30],[32],[5],[8].

Chapter 3

Explanation methods taxonomy

The aim of this work is to present two rather novel XAI techniques called LIME [31] and SHAP [21]. Therefore, in this Chapter, we will overview the different ways of categorizing explanations methods according to their properties and afterwards there will be presented the classes in which these two methods, LIME an SHAP, fit in, in order to get an overall picture of them, that let us explain them in a detailed way in the Sections 5 and 6, respectively.

3.1. Categorization of the methods

In order to "explain"[1] the predictions made by a model, there exist several methods, that can be categorized according to three criteria as can be seen in Table 3.1.

Categorization 1		Categorization 2		Categorization 3	
Intrinsic	**Post-hoc**	**Global**	**Local**	**Model-specific**	**Model-agnostic**

Table 3.1: Categorizations of explanations methods

So, as it can be seen in Table 3.1 in blue, the first categorization distinguishes between intrinsic and post-hoc, setting a differentiation of the point "When?" the intepretability is obtained.

Definition 3.1.1. ***Intrinsic*** *methods (also called Ante-hoc by some authors [8]) are methods that obtain the interpretability at the same time that training the model, by restricting the complexity of the model [26],[30].*

[1] Here "explain" is referred to within inverted commas, because the categorization 1, entails a distinction between *intrinsic* and *post-hoc* methods, but since *intrinsic* means that the model is inherently interpretable, according to the differentiation done in Chapter 2 of the terms *explainable* and *interpretable*, when speaking of *intrinsic* methods, it should be used the term *interpretation* instead of *explanation*.

Definition 3.1.2. ***Post-hoc*** *methods are methods that obtain the interpretability of the model analyzing the model after training it, providing explanations due to its outcomes [8],[30].*

For example, decision trees or linear models are simple models and the interpretability of the models is obtained due to their structure [5],[8]. By contrast, random forests do not have such simple structure, so a post-hoc analysis is needed, in order to interpret, for example, making a feature permutation. Other models that need post-hoc analysis are for example Support Vector Machines or Neural Networks [5].

It follows that achieving the interpretability in an intrinsic way, in fact means that the interpretability relies intrinsically on the structure of the model, i.e. the model is a white-box [12]. Therefore, explainability is grounded on post-hoc interpretability, because the aim is to explain non-interpertable models, i.e. of black-boxes, since "white-boxes" or transparent models are interpretable by their own structure and do not suppose any challenge [5], i.e. when speaking about explanations (and not just interpretation), post-hoc analysis is referred.

The second classification, that can be seen in Table 3.1 in red, refers to the question "What? (is explained by the method)", the whole model or a single prediction.

Definition 3.1.3. ***Global*** *methods are methods that explain the behavior of the model on avarage for a given dataset [24], explaining the whole model [24].*

Definition 3.1.4. ***Local*** *methods are methods that explain the behavior of the model's prediction of a single instance [24].*

So, when getting global explanations, the whole model can be trusted, but when getting local explanations just the single prediction can be trusted [30]. Moreover, global fidelity implies local fidelity, but not the other way around [31], understanding under fidelity the degree of approximation to the behaviour of the model [16],[31]. The problem is that, getting global explanations is not an easy task. That is why, local explanations is the minimum requested when speaking about explanations [31].

At last but not least, the third categorization distinguishes between two kinds of methods, depending on the answer "How? (do they work)".

Definition 3.1.5. ***Model-Specific*** *methods are methods that make the explanations according to the inner structure of the specific model, so they can just be applied to a certain model (class) [12],[30].*

Definition 3.1.6. ***Model-agnostic*** *methods are methods that make the explanations just using the data and the predictions of the model, but not the inner structure of the model, so they can be applied to any model [24], [30] .*

It is worth noting that intrinsic always implies model-specific, and that model-agnostic methods are always post-hoc [26].

A particular techinque of model-agnostic methods is the one based on **surrogate models** (for other techniques see [3]). The surrogate models are interpretable models trained in order to replicate the behavior of the original non-interpretable model, i.e., approximating the predictions made by it, so the explanation consists on interpreting the surrogate model [24], [27].

3.2. Categorization of LIME and SHAP

In what follows, the focus will be set on two particular explanations methods: LIME and some variants of SHAP. Therefore let us first make a classification of these methods according to the terms above explained in Table 3.2.

	Classification 1		Classification 2		Classification 3	
Method	**Intrinsic**	**Post-hoc**	**Global**	**Local**	**Model-specific**	**Model-agnostic**
LIME		✓		✓		✓
Kernel SHAP		✓		✓		✓
Linear SHAP		✓		✓	✓	
Tree SHAP		✓		✓	✓	

Table 3.2: Classification LIME and variants of SHAP according to the properties of Table 3.1

Chapter 4

Additive explanation models

In 2017 Scott M. Lundenberg and Su-In Lee [21] introduced a novel persepective into the area of explaining model predictions "viewing *any* explanation of a model's prediction as a model itself". They termed such explanations as "*explanation model*". Furthermore, they defined a special class of (post-hoc) local explanation methods called *additive feature attribution methods* for methods with linear explanation models, where the explanation model, and therefore the explanation of the prediction is the sum of real values that correspond to each input feature used by the prediction model. In this class fit explanation methods like LIME (see Chapter 5), as well as the one presented in [21] by themselves, SHAP (see Chapter 6).

4.1. Additive feature attribution methods

The aim is explaining **locally** a prediction model via an *explanation model*, i.e. via an interpretable approximation of the prediction model. Let $f : \mathbb{R}^{|N|} \to \mathbb{R}$ be the original prediction model which is wanted to be explained and g the explanation model and let $x \in \mathbb{R}^{|N|}$ be the single instance that wants to be explained (local method), where every dimension of $|N|$ corresponds to an input feature, and $f(x)$ the prediction of the original model of that instance.

Observation 4.1.1. *If f is a classification model, then $f(x)$ represents the probability of belonging to a certain class.*

Next the additive feature attribution methods will be presented, but let us before make a parenthesis in order to note that, although the input that wants to be explained is $x \in \mathbb{R}^{|N|}$, normally, explanation models do not work with x, but with $x' \in \{0,1\}^{|N'|}$ (*note that $|N|$ is not necessary equal to $|N'|$*), called *simplified input* [21] or *interpretable input* [31], distinguishing this way between original and interpretable/simplified representation, where the simplified maps to the original one

through a mapping function $x = h_x(x')$. Whenever $z' \approx x'$, then $g(z') \approx f(h_x(z'))$ (local approximation) [21].

Definition 4.1.1. ***Additive feature attribution methods****[21] are local methods that have an explanation model that is a linear function of binary variables:*

$$g(z') = \phi_0 + \sum_{i=1}^{|N'|} \phi_i z'_i \tag{4.1}$$

where $z' \in \{0,1\}^{|N'|}$, $|N'|$ is the number of simplified input features, and $\phi_i \in \mathbb{R}$ is the effect attribute to each feature, so that the sum of the effect of all features approximates the output of the original model.

Chapter 5

LIME

LIME [31] (Local Interpretable Model-agnostic Explanations) is, as its name implies, a local and model-agnostic method that explains a single prediction $f(x)$ of the model f, using a linear explanation model g such as (4.1) in the proximity of x, i.e. it is assumed local linearity in order to approximate $f(x)$ with the model g.

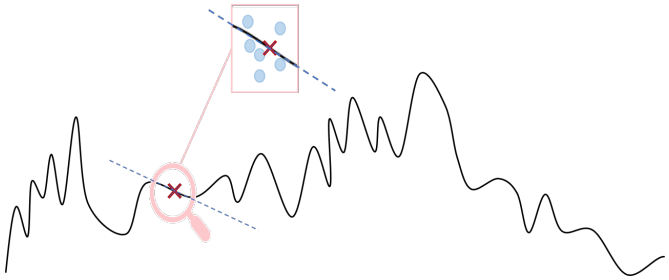


Figure 5.1: Linear approximation in the neighborhood of x

Notation 5.0.1. *In fact, LIME can have as explanation model any interpretable model, not only linear models, but also for example decision trees. Nevertheless, due to the good results the authors obtained in the experiments, explaining black-boxes just with linear explanation models, the authors focused just on this type, leaving the door open to investigate in further work using others.*

5.1. Parameters

The aim now is to determine the weights of g, i.e., returning to (4.1), the values ϕ_i. Therefore the following concepts are introduced:

- G:= **Class of interpretable models** (liner models, decision trees,...). Due to Note 5.0.1, in this work G is the class of linear models.

- $\Omega(g) :=$ **Complexity of** g. This concept is used to penalize g being too complex. As we assume linearity, complexity means number of non-zeros weights. The concrete form selected in the implementation consists in limiting the number of non-zero weights by a constant K.*If K is small, the explanation model is more interpretable, if K is high, the explanation model is more locally faithful.*

- $\pi_x(z)$:= **Local kernel**. Proximity measure between an instance z to the input x. *This parameter is important because we are approximating locally.* Normally, the concrete form used for π_x is:

$$\pi_x(z) = e^{-\frac{D(x,z)^2}{\sigma^2}} \tag{5.1}$$

  where D is a measure of distance between x and z (depending on the type of input data, that can be text, images or tabular data, D is the cosine distance (text) or the euclidean distance (otherwise)) and σ is the kernel-width [4] (normally $\sigma = \frac{3}{4}\sqrt{|N|}$ for tabular data and 25 for textual data [4]). The value of σ is responsable for the local fidelity, because, if it is too large, the weight given for different instances is very similar (i.e., close and far away instances have the same weight), but if it is too small, just a few points will be weighted high (i.e., just the ones too close have weight)

- $\mathcal{L}(f, g, \pi_x)$:= **Locality-aware loss**. Measure of how unfaithful is g in aproximating f locally using π_x. The used form of $\mathcal{L}$ is the locally weighted square loss:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{5.2}$$

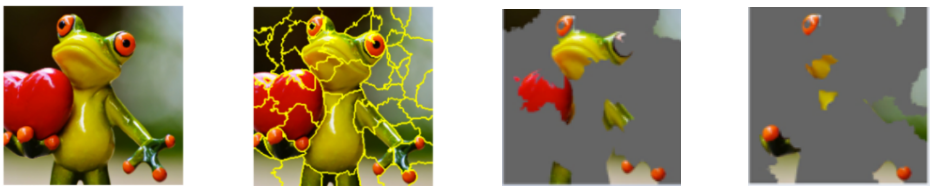
5.2. Sampling

As it can be seen in Equation (5.2), $\mathcal{L}$ depends on the elements $z \in \mathcal{Z}$ and their simplified representation, where $\mathcal{Z}$ is a dataset of perturbations of the instance that needs to be explained (x or its simplified representation x'). The use of $\mathcal{Z}$ permits to approximate the locality-aware loss, in a model-agnostic way, making no assumptions of the original model f, because it let us know the local behavior of f, due to the fact that, the model is a black box which inner working is not known, but the predictions of other instances z can be obtained. This way, $\mathcal{L}$ is set in Equation (5.2), as the difference between the predictions of f and the explanation model g for perturbed

instances, weighting the distance between instances with π_x, so that the ones in the neighborhood are given a higher weight.

The way of getting the sample depends on the type of data available: Images, text or tabular data. Let us see all three cases, and therefore the form of the original and the simplified form of the data.

1. **Images**. Since in images the original representation x is given by the pixels of the image (see Figure 5.2), this gives no much information, because isolated pixels do not have any meaning. So, the simplified representation x' is given by "super-pixels"(see Figure 5.2(b)), i.e. by groups of pixels (grouped by the Quick-Shift algorithm, although other algorithms could be used [35]), where 0 means that the super-pixel is off (shaded gray in the image) and 1 means it is on. Notice that here the dimension of x', $|N'|$, is smaller than the one of x, $|N|$.
   Therefore, in order to obtain the dataset of perturbed samples $\mathcal{Z}$, it is sampled around the simplified instance x' drawing non-zero elements of x' uniformly at random (the number of the draws is also uniformly sampled), so that the non-zero elements of the sampled input z' form a subset of non-zero elements of x'(see Figures 5.2(c),5.2(d)). Mapping the sample to the original representation $z = h_x(z')$, the prediction $f(z)$ as well as the weight $\pi_x(z)$ are then obtained. In the end, a whole dataset $\mathcal{Z}$ and the corresponding predictions and weights are gotten.



(a) Original input (b) Simplified input (c) Perturbation (d) Perturbation

Figure 5.2: Scheme of sampling images.[1]

2. **Textual data**[23]. When using textual data, the original representation x are word embeddings. Suppose in x there are $|N'|$ distinct words and denote them by $\{w_1, ..., w_{|N'|}\}$. The first step is to sample uniformly at random in $\{1, ..., |N'|\}$ a number s. Then a subset of size s, $S \subseteq \{1, ..., |N'|\}$, is sampled uniformly at random. Then we get a new sampled instance z, by suppressing in x all the words

[1]Images by Marco Tubeiro, Sameer Singh and Carlos Guestrin (authors of the original paper of LIME [31]) in `https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/`

Instance in original space	Simplified instance
She works as data scientist	(1,1,1,1,1)
She as scientist	(1,0,1,0,1)
works as	(0,1,1,0,0)

Table 5.1: Example sampling text data

$w_i, \forall i \in S$. The simplified instance z' is then the binary vector, with $z'_i = 1$ if the word w_i of x appears in z, and $z'_i = 0$ otherwise. Therefore, the simplified input of x is $x' = (1, ..., 1)$. This procedure is done until a whole dataset $\mathcal{Z}$ is gotten. An example is given in Table 5.1, where the given instance is "She works as data scientist", and the resulting perturbations are "She as scientist" and "works as".

3. **Tabular data**. When using tabular data it must be distinguished between quantitative or categorical features. Nevertheless, both have in common, in contrast to images and textual data, that here a training set is required.

   - **Categorical**[2]. When a feature is categorical, the sampling is bounded to the frequency of the different classes of the training dataset. Say the $i-th$ feature is categorical. So x_i corresponds to a class of the possible ones. The simplified input takes the value 1, i.e $x'_i = 1$. It will be then sampled x_i according to the frequency of each class obtaining different z_i. If $z_i = x_i$, i.e. the sample corresponds to the same class, then the simplified input is $z'_i = 1$, otherwise it is $z'_i = 0$.
     For example, suppose the $i-th$ feature is *Marital status* and with the three following classes: *Married, Single* and *Divorced*, with frequencies $0.5, 0.2$ and 0.3, respectively. Say x_i =*Married*, so $x'_i = 1$ and any $z'_i = 1$ just if z_i =*Married*. Sampling (for example 5 times) we get $z_i^1, ..., z_i^5 \in \mathcal{Z}$ and their simplified inputs like in Table 5.2.

[2]Information given by Python with `help(lime.limetabular)` after `from lime.limetabular import LimeTabularExplainer`

Pertubed instances in original space (z_i)	Perturbed instances in simplified space (z'_i)
Divorced	1
Married	0
Single	0
Single	0
Divorced	1

Table 5.2: Example sampling categorical feature

- **Quantitative**. Here we may distinguish two forms of sampling. The first one, and recommended by the author, is discretizing the feature into he empirical quantiles of the training set. The second one, consists in using the original instance as the simplified instance itself (this time the simplified representation is not binary).

  a) **Discretizing**[15]. When a feature is numerical, the feature is discretized in agreement to the empirical quantiles of the training set. The idea is to establish categorical features from the quantitative ones. Therefore, let the $i-th$ feature be quantitative. If x_i has a value between $\hat{q}_k$ and $\hat{q}_{k+1}$, then the discrete representation of x_i is k, so that the simplified input is $x'_i = 1$ and is interpreted as x_i discretized is k. Suppose there are in total p quantile-boxes (i.e bins picked by the quantiles). Next step is to sample uniformly $\{1, ..., p\}$. Let s denote the value obtained. By sampling from a normal distribution (with mean and standard deviation of the distribution of the training dataset) truncated to the corresponding quantile-box s we get the $i-th$ feature of the searched sampled instances z_i. The simplified representation z'_i takes the value 1 if z_i falls into the same quantile-box as x_i, and 0 otherwise. Naturally, this is done for every quantitative feature in x.
  For example, suppose the $i-th$ feature is *Age* and the discretization is done into quartiles, so $p = 4$. Let the quartiles be given, for example, by $q_1 = 0, q_2 = 15, q_3 = 39, q_4 = 56, q_5 = 100$ (see the notation is the above explained and not the usual one for the indexes of quartiles), and the original feature $x_i = 25$. The simplified input is $x'_i = 1$, meaning that $x_i \in [q_2, q_3)$ and every perturbed instance z'_i is 1 only if z_i belongs to the interval with extreme points $q_2 = 15$ and $q_3 = 39$, as x_i, and 0 otherwise.

q_1	q_2	q_3	q_4	q_5
0	15	39	56	100

instance in original space	**instance in simplified space**
14	0
48	1
32	1
91	0
56	0

Table 5.3: Example sampling quantitative feature discretized

b) **Without discretizing**. If no discretizing is done, then the simplified and the original representation of the $i-th$ quantitative feature coincide ($x_i = x'_i$)[3]. So, the perturbation is done, sampling from a $Normal(0,1)$ and multiplying by the standard deviation and adding back the mean (with mean and standard deviation of the distribution of the training dataset). Nevertheless, this is not recommended by the author of [31], Marco Tubeiro, because, the interpretation becomes harder, since for example negative weights for negative features mean positive contribution (double negatives are hard to think about).[4]

5.3. Optimization Problem

Once all above concepts are explained, let us expose how LIME gets the explanation model. To find the values ϕ_i, i.e., to determine g, LIME considered the following optimization problem:

$$\arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{5.3}$$

so that, the explanation model g that verifies Eq.(5.3), is the local explanation of $f(x)$.

Due to the parameters selected in Section 5.1, solving the Equation (5.3) is intractable. Therefore, the procedure used to do this minimization as an approximation, called in [31] as *K-LASSO*, consists in fixing a number K number of non-zero weights ϕ_i that should have the model and using LASSO regularization to select the corresponding K features. Then the weights are learned via least squares. This way, going back to Section 4.1, the weights here calculated of the explanation model represent the effect attribute to each feature, so that, we have got an explanation of the prediction $f(x)$ that is interpretable.

[3]This is explained in the official blog of the author of [31] in `https://github.com/marcotcr/lime/issues/485`

[4]This is expressed by Marco Tubeiro, author of [31], on the page `https://github.com/marcotcr/lime/issues/196`, that corresponds to the comments of the author on the repository of the lime package.

Chapter 6

SHAP

Another additive feature attribution method is SHAP (SHapley Additive exPlanations) [21]. This method is based on Game Theory concepts, specifically, on the Shapley values. So let us first explain this concept.

6.1. Shapley Values

Shapley values [1] were introduced 1953 by Lloyd Shapley as part of the cooperative Game Theory in an axiomatic ways, i.e. certain properties were proposed and it was proved that only one value concept, the Shapley values, satisfied all properties simultaneously, becoming the unique method to attribute **fairly** the payoff of a cooperative game to the players. Nevertheless, in order to make it more comprehensible let us see first the definition of the Shapley values and then the properties fulfilled.

Definition 6.1.1. *Let N' be a set of players of a game, S a subset of players, i the $i-th$ player and $v : 2^{N'} \to \mathbb{R}$ the set function, i.e. the worth of S, with $v(\emptyset) = 0$, then the Shapley value of the $i-th$ player is:*

$$\phi_i(v) = \sum_{S \subseteq N' \setminus \{i\}} \frac{|S|!(|N'|! - |S| - 1)!}{|N'|!}(v(S \cup \{i\}) - v(S)) \tag{6.1}$$

The term $\frac{|S|!(|N'|!-|S|-1)!}{|N'|!}$ indicates the probability of the $i-th$ player to incorporate to S, while the term $(v(S \cup i) - v(S))$ is the marginal contribution of the $i-th$ player when it incorporates to S. Therefore, the $i-th$ Shapley value is the average of the marginal contribution of the $i-th$ player, over all the possible permutations the $i-th$ player can be part of (i.e. the order matters).

The properties that are only satisfied by Shapley values simultaneously are following:

Property 6.1.1. ***Efficiency*** *The total payoff of the game is shared among all $|N'|$ players:*

$$v(N') = \sum_{i \in N'} \phi_i(v)$$

Property 6.1.2. ***Symmetry*** *If players $i \in N'$ and $j \in N'$ are symmetric, i.e. $v(S \cup \{i\}) = v(S \cup \{j\})\ \forall S \subseteq N \setminus \{i, j\}$, then their payoff are the same: $\phi_i(v) = \phi_j(v)$.*

Property 6.1.3. ***Additivity*** *If w is another set function, the payoff of the sum-game is the sum of the payoffs of the single games:*

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w)$$

Property 6.1.4. ***Null-player*** *If the $i - th$ player is a null player, i.e., $v(S \cup \{i\}) = v(S)\ \forall S \subset N', i \in S$, then $\phi_i(v) = 0$*

Let us see an example of Shapley values.

Example 6.1.1. *Let $N' = \{1, 2, 3\}$ be a set of 3 players, and the set function taking following values:*

$$v(S) = \begin{cases} 0 & \text{if } S = \emptyset \\ 100 & \text{if } S = \{1\} \\ 125 & \text{if } S = \{2\} \\ 50 & \text{if } S = \{3\} \\ 250 & \text{if } S = \{1, 2\} \\ 350 & \text{if } S = \{1, 3\} \text{ or } S = \{2, 3\} \\ 500 & \text{if } S = \{1, 2, 3\} \end{cases}$$

Depending on the moment the player i comes to the coalition, the different possible ordering of players are

1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1

So, according to these possible orderings, the marginal contributions are the ones of Table 6.1.

Marginal contribution	i	1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
$v(S \cup \{i\}) - v(S)$	1	100	100	125	150	300	150
	2	150	150	125	125	150	300
	3	250	250	250	225	50	50

Table 6.1: Marginal contributions

Therefore, the Shapley values are:

$$\begin{aligned}
\phi_1 &= \sum_{S \subseteq N' \setminus \{1\}} \frac{|S|!(|N'|! - |S| - 1)!}{|N'|!}(v(S \cup \{1\}) - v(S)) \\
&= \frac{0!(3-0-1)!}{3!}(\underbrace{v(\emptyset \cup \{1\})}_{100} - \underbrace{v(\emptyset)}_{0}) + \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{2\} \cup \{1\})}_{250} - \underbrace{v(\{2\})}_{125}) \\
&+ \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{3\} \cup \{1\})}_{350} - \underbrace{v(\{3\})}_{50}) + \frac{2!(3-2-1)!}{3!}(\underbrace{v(\{2,3\} \cup \{1\})}_{500} - \underbrace{v(\{2,3\})}_{350}) \\
&= \frac{1}{6}(2100 + 125 + 300 + 2150) = \frac{925}{6}
\end{aligned}$$

$$\begin{aligned}
\phi_2 &= \sum_{S \subseteq N' \setminus \{2\}} \frac{|S|!(|N'|! - |S| - 1)!}{|N'|!}(v(S \cup \{2\}) - v(S)) \\
&= \frac{0!(3-0-1)!}{3!}(\underbrace{v(\emptyset \cup \{2\})}_{125} - \underbrace{v(\emptyset)}_{0}) + \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{1\} \cup \{2\})}_{250} - \underbrace{v(\{1\})}_{100}) \\
&+ \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{3\} \cup \{2\})}_{350} - \underbrace{v(\{3\})}_{50}) + \frac{2!(3-2-1)!}{3!}(\underbrace{v(\{1,3\} \cup \{2\})}_{500} - \underbrace{v(\{1,3\})}_{350}) \\
&= \frac{1}{6}(2125 + 150 + 300 + 2150) = \frac{1000}{6}
\end{aligned}$$

$$\begin{aligned}
\phi_3 &= \sum_{S \subseteq N' \setminus \{3\}} \frac{|S|!(|N'|! - |S| - 1)!}{|N'|!}(v(S \cup \{3\}) - v(S)) \\
&= \frac{0!(3-0-1)!}{3!}(\underbrace{v(\emptyset \cup \{3\})}_{50} - \underbrace{v(\emptyset)}_{0}) + \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{1\} \cup \{3\})}_{350} - \underbrace{v(\{1\})}_{100}) \\
&+ \frac{1!(3-1-1)!}{3!}(\underbrace{v(\{2\} \cup \{3\})}_{350} - \underbrace{v(\{2\})}_{125}) + \frac{2!(3-2-1)!}{3!}(\underbrace{v(\{1,2\} \cup \{3\})}_{500} - \underbrace{v(\{1,2\})}_{250}) \\
&= \frac{1}{6}(250 + 250 + 225 + 2250) = \frac{1075}{6}
\end{aligned}$$

And it can be seen that $\phi_1 + \phi_2 + \phi_3 = \frac{925+100+1075}{6} = 500 = v(\{1,2,3\})$.

6.2. Connection between Shapley values and additive feature attribution methods

To establish a connection between this Game Theory concept and the additive feature attribution methods, the payoff of the game must be understood as the model prediction, and the players as the input features. So, the total payoff, i.e., the predicted value of an instance x, is sought to be distributed in a *fair* way, respecting Properties 6.1.1,6.1.2,6.1.3 and 6.1.4. Therefore, the simplified inputs $z' \in \{0,1\}^{|N'|}$, must be understood as a representation of a coalition $S \subseteq N'$, so that, if an entry is $z'_i = 1$ it means that the $i-th$ feature is present, and if it is $z'_i = 0$, the features is absent.

Game Theory	Additive feature attribution method
Payoff (v)	Prediction ($f(x)$)
Players (N')	Features (N')
Coalition of players (S)	Coalition of features (z')
Shapley values (ϕ)	Contribution of each feature to the prediction (ϕ)

Table 6.2: Connections between Game Theory and additive explanation methods of black boxes

6.3. SHAP

SHAP [21],[20] arises of the fact that there is only a unique solution in the class of additive feature attribution methods satisfying the three following Properties 6.3.1,6.3.2 and 6.3.3.

Property 6.3.1. ***Local accuracy***

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{|N'|} \phi_i x'_i \tag{6.2}$$

Property 6.3.1 expresses that the sum of the feature attributions and the output of the original model agree. This property agrees with the property of efficiency of the Shapley values (Property6.1.1).

Property 6.3.2. ***Missingness***

$$x'_i = 0 \Rightarrow \phi_i = 0 \tag{6.3}$$

Property 6.3.2 indicates that, if a feature is missing , the feature attribution is zero, i.e., it is given no importance.

Property 6.3.3. ***Consistency*** *Let $f_x(S) = f(h_x(z'))$ where S is the set of non-zero indexes in z'. For any two models f and f':*
if $f'_x(S) - f'_x(S \setminus \{i\}) \leq f_x(S) - f_x(S \setminus \{i\}) \, \forall S \subseteq N'$, then $\phi_i(f', x) \leq \phi_i(f, x)$.

Property 6.3.3 states that, if a model is changed, increasing the impact of a feature, the attribution given to that feature will not decrease. In the appendix of [21] it is exposed that this property implies the properties of linearity, null-player and symmetry of the Shapley values (Properties 6.1.3,6.1.4 and 6.1.2).

6.3.1. Unique solution

That unique solution of the class of additive feature attribution methods , satisfying Properties 6.3.1,6.3.2 and 6.3.3 is given by:

$$\phi_i(f, x) = \sum_{S \subseteq N' \setminus \{i\}} \frac{|S|!(|N'| - |S| - 1)!}{|N'|!}(f_x(S \cup \{i\}) - f_x(S)) \tag{6.4}$$

where S are all the possible coalitions of features (players as denoted in Section 6.1) in N' not containing the $i - th$ feature and $|S|$ its size.[1]

These values ϕ_i are the Shapley values seen in Section 6.1. Hence, Game Theory and additive feature attribution methods unify and present the *unique* additive feature attribution method that is locally accurated and consistent, respecting also missingess.

[1]Due to the fact that $f_x(S) = f(h_x(z'))$ in some papers like [21] it is used the notation $f_x(z') = f(h_x(z'))$ instead the above mentioned. It is worth mentioning it because the relation between both notations emphasizes that the features forming the coalition S are the non-zero entries in the simplified input z', and therefore the size of S is the same as the number of non-zero entries in z' that can be noted as $|z'|$.

6.3.2. SHAP values

The SHAP values [21] are defined as the Shapley values of a conditional expectation function of the original model, where:

$$f_x(S) = f(h_x(z')) = E(f(z)|z_S = x_S) \tag{6.5}$$

This way, due to the fact that Shapley values can refer to any set function, the term SHAP values is used when refering to the specific case when the set function is set as the conditional expectation. Behind this definition is that $h_x(z') = z_S$, but z_S has missing values for the features not in S. For example, suppose the features are $N' = \{$ age, high, weight $\}$, and $S = \{$ age, high $\}$. Then, z_S would be of the form $z_S = (20, 1.75, 55,\text{missing})$. Therefore, in order to solve the problem of missing values, $f_x(S) = f(h_x(z')) = f(z_S)$ is approximated with $E(f(z)|z_S = x_S)$. Nevertheless, computing these values is too effortful. Therefore, the SHAP values can be approximated assuming feature independence to the marginal expectation as follows [2]:

$$\begin{aligned} f_x(S) =& f(h_x(z')) = E(f(z)|z_S = x_S) = E(f(z_{\bar{S}}, z_S)|z_S = x_S) \\ &= \int f(z_{\bar{S}}, x_S)p(z_{\bar{S}}|z_S = x_S)dz_{\bar{S}} \\ &\underbrace{\approx}_{\text{feature independece}} \int f(z_{\bar{S}}, x_S)p(z_{\bar{S}})dz_{\bar{S}} = E(f(z_{\bar{S}}, z_S)). \end{aligned} \tag{6.6}$$

Moreover, assuming model linearity simplifies the expression to:

$$f_x(S) = f(h_x(z')) = E(f(z)|z_S = x_S) \approx E(f(z_{\bar{S}}, z_S)) \underbrace{\approx}_{\text{linearity}} f(E(z_{\bar{S}}), z_S). \tag{6.7}$$

Summing up, SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature, i.e., they indicate how to get from $E(f(z))$ (called *base value*) to the prediction $f(x)$, by summing the SHAP values. For example, see Figure 6.1, and suppose we have got 3 features. In a) we see the 0, the $E(f(z))$ and the prediction $f(x)$. The value of $E(f(z))$, as shown in b), is ϕ_0. If we add ϕ_1 like in c), we end in $E(f(z)|z_1 = x_1)$, i.e. $\phi_1 = E(f(z)|z_1 = x_1) - E(f(z))$ (here $S = feature1$). In d) is also added $\phi_2 = E(f(z)|z_{1,2} = x_{1,2}) - E(f(z)|z_1 = x_1)$, and at last at e), also $\phi_3 = E(f(z)|z_{1,2,3} = x_{1,2,3}) - E(f(z)|z_{1,2} = x_{1,2})$ is added, ending at $f(x)$, i.e., the sum of the ϕ_i gives the difference between $E(f(z))$ and $f(x)$.

In Figure 6.1 just an order of adding the features is exposed. Nevertheless, if the features are not independent or the model is non-linear, the order of adding features matters, so the SHAP values are obtained averaging the ϕ_i over all orderings.

Notation 6.3.1. *Even though the marginal expectation is obtained by simplification assuming feature independence, in [18] it is shown, making use of causal inference that, in fact, the marginal expectation of Equation (6.6) is conceptually the right one, and not an approximation, differentiating observational conditional expectation (the one of Equation (6.5)) and what they call interventional conditional expectation that leads to the marginal expectation. The original authors of SHAP [21], deepen in [11] into this theme and say that, in general, there neither of the two options is the right one, but it can be chosen the one or the other based on the application.*

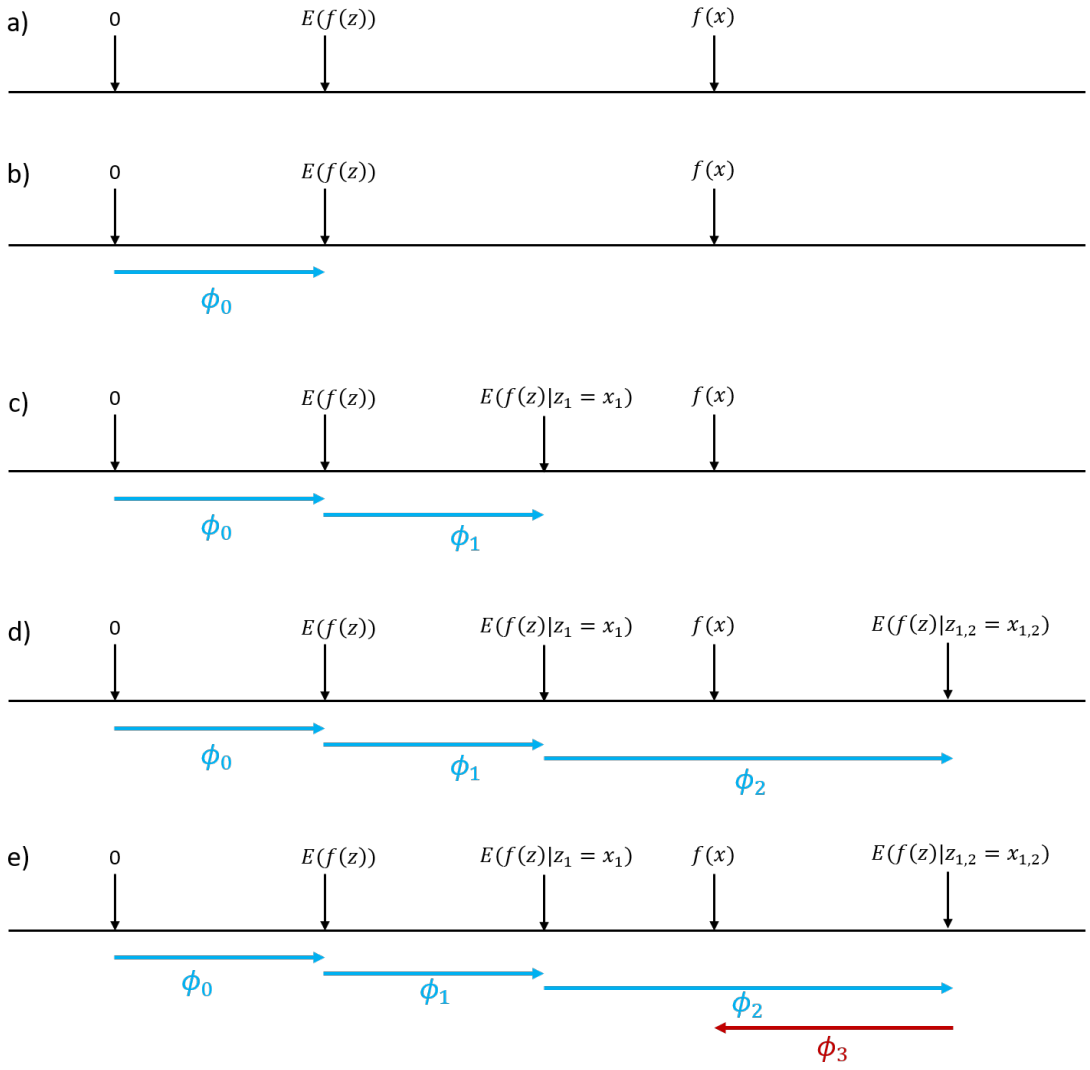


Figure 6.1: SHAP values scheme

6.4. KernelSHAP

KernelSHAP [21] is a model-agnostic method that combines LIME and Shapley values. On the one hand, it is known that LIME (Chapter 5) finds an explanation minimizing equation (5.3) and it is also known that LIME is an additive feature method. On the other hand, it is also known that the unique solution fulfilling (4.1), and Properties 6.3.1,6.3.2,6.3.3 are the Shapley values. Therefore, Shapley values are the only possible solution of (5.3), satisfying Properties 6.3.1,6.3.2,6.3.3. Nevertheless, it must be done the right choice of the terms in (5.3), in order this equation recovers the Shapley values. This choice is given, instead of the values used by LIME in Chapter 5, by:

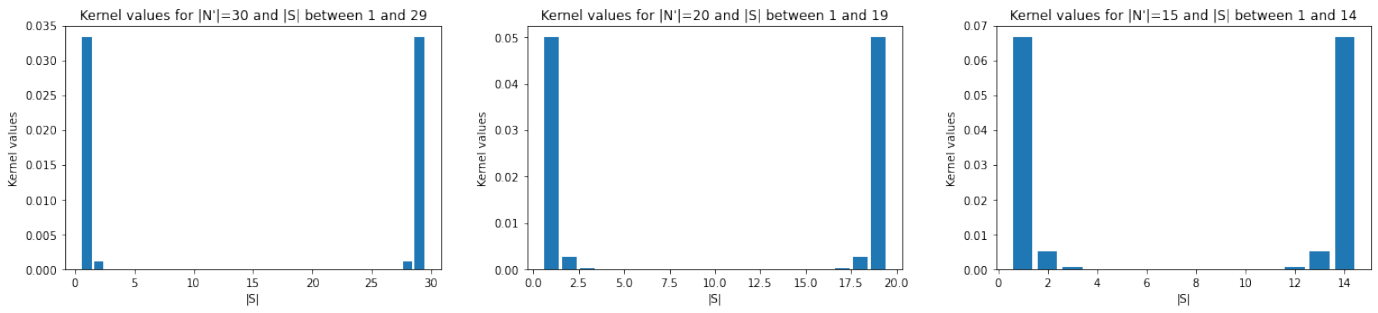
$$\Omega(g) = 0 \tag{6.8}$$

$$\pi_{x'}(z') = \frac{|N'| - 1}{\binom{|N'|}{|S|}|S|(|N'| - |S|)} \tag{6.9}$$

where, $S \subseteq N'$ is the set non-zero features in z'.

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in \mathcal{Z}} (f(h_x(z')) - g(z'))^2 \pi_{x'}(z') \tag{6.10}$$

As it can be seen above, $\mathcal{L}$ is calculated like in LIME over a set $\mathcal{Z}$. The reason is that the "best" would be to calculate it all over N', i.e. making all possible coalitions. Nevertheless, the power set of N' is much too large. Therefore, KernelSHAP samples coalitions of N' according to the probability distribution induced by the kernel (Eq. 6.9) [18]. In Figure 6.2 it can be observed for different values of $|N'|$ that, the Kernel only takes high values when $|S|$ is next to 0 or to $|N'|$.



(a) $|N'| = 30$ (b) $|N'| = 20$ (c) $|N'| = 15$

Figure 6.2: Kernel values for different $|N'|$

It is also worth mentioning that, when the coalition has not non-zero entries or all entries are non-zero, i.e., if $|S| = 0$ or $|S| = |N'|$, then $\pi_x(z') = \infty$ (this is the reason why these values are not represented in Figure 6.2). Therefore, it is necessary to impose the constraints $\phi_0 = f_x(\phi)$ and $f(x) = \sum_{i=0}^{|N'|} \phi_i$.

Moreover, going back to Equation (6.6), Kernel SHAP approximates the integral using the empirical distribution of the training set used to train the original model f, by [2]:

$$f_x(S) = f(h_x(z')) \approx \frac{1}{J} \sum_{j=1}^{J} f(z_{\bar{S}}^j, x_S) \tag{6.11}$$

where $z_{\bar{S}}^k$, $j = 1, .., J$ are samples of the training set.

The optimization problem is then solved using weighted least squares.

Notation 6.4.1. *In LIME it was used as kernel function $\pi_x(z)$, but here the notation is replaced by $\pi_{x'}(z')$. This occurs because in LIME the distance between two instances is calculated through the euclidean or the cosine distance, measured therefore in the original space, meanwhile in KernelSHAP measures the proximity between instances as proximity between coalitions in the simplified space. Nevertheless, both refer to the proximity between the two instances, so it is just a notation question. In fact, [21], notes both as $\pi_{x'}(z')$.*

6.5. TreeSHAP

TreeSHAP [20],[22] is a model specific method for computing the exact SHAP values for tree-based models. The TreeSHAP, does not calculate $f(h_x(z'))$ as the approximate marginal expectation, but as the conditional expectation. Originally this algorithm derives from an algorithm that calculates the conditional expectation $E(f(z)|z_S = x_S)$ in $O(TL|N'|2^{|N'|})$, ending up into a polynomial algorithm of $O(TLF^2)$, where T is the number of trees, L the number of terminal nodes (leaves), F the maximum depth of any tree and $|N'|$ the number of features [22].

The algorithm of $O(TL|N'|2^{|N'|})$ follows the next idea: First of all, Shapley values are additive (Property 6.1.3), so when the tree-based model is a tree ensemble, the SHAP values (remember they are Shapley values) are the average of the SHAP values of the single trees. Secondly, the SHAP values for every single tree can be calculated, calculating $E(f(z)|z_S = x_S)$ as follows. Due to the fact that the only known values of z are $z_S = x_S$, when z goes down the tree, i.e. when the tree is getting a prediction of z, it can fall in several terminal nodes. Therefore, the scheme to obtain $E(f(z)|z_S = x_S)$ is: if the split feature is in S, then follow the decision path of z, and if the split feature

is not in S, then take the weighted (due to the number of samples of the training set matching the conditioning set S, that fall into each node, when going down the tree) average of both branches. Hence, if S is formed by all features ($|S| = |N'|$), the expected value will coincide with the value of the only terminal node into which z can fall; if S is formed by non features ($|S| = 0$), the expected value will be the weighted average of all terminal nodes; and at least, if S is formed just by some features ($|S| \neq 0, |N'|$), the expected value will be the weighted average of the terminal nodes that can be reached by z due to $z_S = x_S$, so that there is no contradiction in the splits that end up to that terminal nodes with the values of z_S.

The improved algorithm that runs in $O(TLF^2)$ used by TreeSHAP is similar to run the $O(TL|N'|2^{|N'|})$ algorithm simultaneously for all the subsets of S. For an indeep review of the algorithm the reader is referred to [20] and [22].

6.6. Other variants

The above exposed variants of SHAP are the most used, KernelSHAP because it is model agnostic and TreeSHAP because it deals with tree ensembles. Nevertheless, there are other model-specific variants of SHAP like LinearSHAP which is used to explain linear models, or DeepSHAP, used to explain DNN-models as exposed in [21].

6.7. SHAP interaction values

The Shapley values extend to the Shapley interaction index[13]. Same way the SHAP values extend to the *SHAP interaction values* [20],[22], which represent the local interaction effects by a matrix of size $|N'| \times |N'|$, where the main effects are on the diagonal and the interaction effects on the other elements of the matrix. They represent the difference between the SHAP values for the $i-th$ features when the $j-th$ feature is present and absent. They are calculated as:

$$\Phi_{i,j} = \sum_{S \subseteq N' \setminus \{i,j\}} \frac{|S|!(|N'| - |S| - 2)!}{2(|N'| - 1)!} \nabla_{i,j}(S),\ i \neq j \tag{6.12}$$

where

$$\nabla_{i,j}(S) = f_x(S \cup \{i,j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) + f_x(S) \tag{6.13}$$

The main effects for the prediction are defined as the difference beteween the SHAP values and the SHAP interaction values:

$$\Phi_{i,i} = \phi_i - \sum_{i \neq j} \Phi_{i,j}. \tag{6.14}$$

6.8. Comparing LIME-SHAP

The main differences between LIME and SHAP are:

1. LIME is model-agnostic whereas SHAP has a model-agnostic implementation (KernelSHAP) but also model-specific variants (TreeSHAP, LinearSHAP, DeepSHAP etc.)

2. SHAP values are fair, guaranteeing perfectly distributed effects.

3. SHAP values explain the difference between the prediction an the global average prediction, while LIME explains the difference between the prediction and the local average prediction [2].

Moreover when focusing on LIME and KernelSHAP, the parameters used in LIME are heuristically chosen while in KernelSHAP they are chosen in order to recover the Shapley values. Therefore complexity measure used in KernelSHAP is set to zero, and the proximity measure does not use a distance measure as in LIME, but measures the proximity according to the features "present" and "absent" in the coalitions.

Nevertheless, both methods are additive feature attribution methods that explain locally the prediction model, giving an explanation that is interpretable. An implementation of both methods is exposed in Chapter 7.

Chapter 7

Applications explanation models

In this Chapter, the already seen methods will be used with four tabular data sets, called `SouthGermanCredit`[1], `COMPAS`[2], `growth`[3] and `tecator`[4], the last two, of functional data.

The procedure will be as follows with the different data sets: The available data will be divided in 70% training set, 30% test set. With the training set, the prediction model, a random forest, will be constructed. Then it will be aimed to explain according to LIME, KernelSHAP and TreeSHAP the predictions done by the forest of instances in the test set. The whole implementation is done in **Python**.

7.1. SouthGermanCredit

The first database under study is `SouthGermanCredit`. This data set consists of 1000 instances, that correspond to 1000 credits from a bank in southern Germany, with a binary response variable, that determines if the credit risks are good (1) or bad (0), and 20 explanatory variables, 3 of them quantitative and 17 categorical. The explanation variables are shown in Table7.1.

Table 7.1: Explanation variables of `SouthGermanCredit`

Variable	Description	Classes (when categorical)
laufkont	status	1 : no checking account 2 : ... <0 DM 3 : 0<= ... <200 DM 4 : ... >= 200 DM / salary for at least 1 year

[1] Available on the UCI Machine Learning Repository
[2] Available on `https://github.com/propublica/compas-analysis`
[3] Available in `R` in the `fda.usc` package
[4] Available in `R` in the `fda.usc` package

Variable	Description	Classes (when categorical)
laufzeit	duration	—
moral	credit history	0 : delay in paying off in the past 1 : critical account/other credits elsewhere 2 : no credits taken/all credits paid back duly 3 : existing credits paid back duly till now 4 : all credits at this bank paid back duly
verw	purpose	0 : others 1 : car (new) 2 : car (used) 3 : furniture/equipment 4 : radio/television 5 : domestic appliancesw 6 : repairs 7 : education 8 : vacation 9 : retraining 10 : business
hoehe	amount	—
sparkont	savings	1 : unknown/no savings account 2 : ... <100 DM 3 : 100 <= ... <500 DM 4 : 500 <= ... <1000 DM 5 : ... >= 1000 DM
beszeit	employment duration	1 : unemployed 2 : <1 yr 3 : 1 <= ... <4 yrs 4 : 4 <= ... <7 yrs 5 : >= 7 yrs
rate	installment rate	1 : >= 35 2 : 25 <= ... <35 3 : 20 <= ... <25 4 : <20
famges	marital status and sex	1 : male : divorced/separated 2 : female : non-single or male : single 3 : male : married/widowed 4 : female : single
buerge	other debtors	1 : none 2 : co-applicant 3 : guarantor
wohnzeit	present residence	1 : <1 yr 2 : 1 <= ... <4 yrs

Variable	Description	Classes (when categorical)
		3 : 4 <= ... <7 yrs 4 : >= 7 yrs
verm	property	1 : unknown / no property 2 : car or other 3 : building soc. savings agr./life insurance 4 : real estate
alter	age	—
weitkred	other installment plans	1 : bank 2 : stores 3 : none
wohn	housing	1 : for free 2 : rent 3 : own
bishkred	number of credits	1 : 1 2 : 2-3 3 : 4-5 4 : >= 6
beruf	job	1 : unemployed/unskilled - non-resident 2 : unskilled - resident 3 : skilled employee/official 4 : manager/self-empl./highly qualif. employee
pers	people liable	1 : 3 or more 2 : 0 to 2
telef	telephone	1 : no 2 : yes (under customer name)
gastarb	foreign worker	1 : yes 2 : no

With the above explained dataset a classification random forest is trained. Next step is to get the explanations for the predictions that the forest does for single instances of the test set. It is worth mentioning before going on that the whole analysis is done from the point of view of "probability of classifying as class 1", i.e. $f(x)$ is the probability of classifying x as class 1.

Figures 7.3 to 7.8 expose the results for five different instances of the test set, explained by KernelSHAP, TreeSHAP and LIME. The SHAP graphics show the base value, the output value and the SHAP values of the different features sorted into two colors, pink and blue, depending on if the feature pushes (like a force -this type of plot is called force plot-) up the output value (SHAP value positive) from the base value, i.e. forces to class 1, or if it pushes down the output value (SHAP value negative), i.e. forces to class 0, respectively. Additionally, they are ordered from bigger to lower con-

tribution. The LIME graphic shows in `Right`, the real prediction of the forest ($f(x)$), in `Intercept` the intercept ϕ_0 of the explanation model g and in `Prediction local` the prediction done by g, that is the sum of the intercept and the contribution values attributed to each feature ϕ_i, and that can be seen on the central graphic. The features in the graphic are sorted from top to bottom, from bigger effect to lower effect. The ones in orange contribute pushing the prediction up, and the blue ones down, and therefore, the corresponding contribution values must be set negative when summing all ϕ_i to the intercept ϕ_0, to get the local prediction. On the table it can be seen the values of the features. It is worth to mentioning that the quantitative features have been discretized into quartiles.

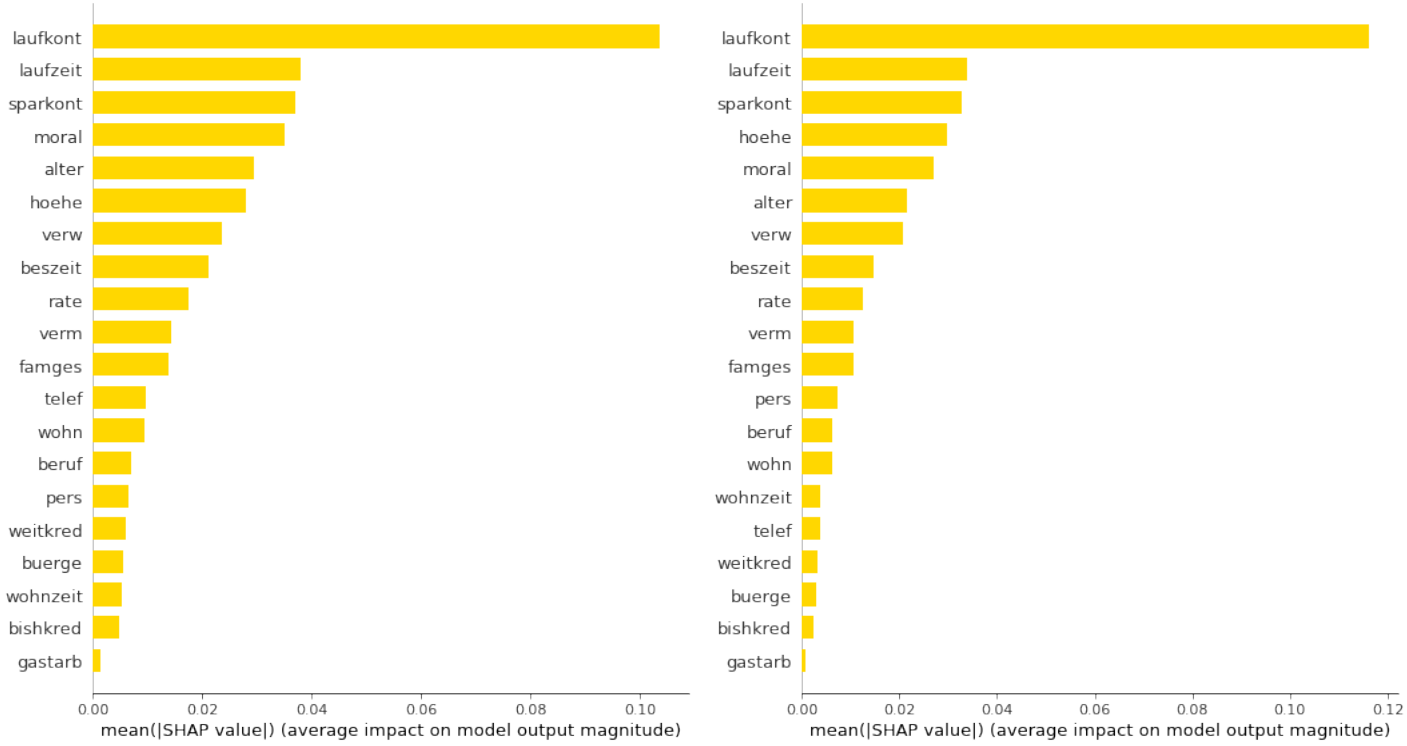
So let us analyze in deeper the results given in Figures 7.4 to 7.8. For every given instance, the results offered by all three explanation methods are very similar, although they differ at some aspects.

LIME for all the displayed instances attributes the feature `laufkont` the highest effect on the prediction, making the class 4 (status $\geq$ 200 DM) responsible for pushing the prediction to class 1 (Figures 7.3 and 7.4), and classes 1 and 2 (no checking account or status $< 0DM$, respectively) responsible for pushing the prediction down to class 0 (Figures 7.5,7.6,7.7 and 7.8). The immediately features with the highest effects are `sparkont` and `laufzeit`. In the case of `sparkont`, the classes 4 and 5 (savings between 500 and 1000 DM, and savings over 1000 DM, respectively) rise the output value to class 1 (Figures 7.3 and Figures 7.6), while class 1 (unknown or no savings) implies a lowering of the output value (Figures 7.4,7.5 and 7.7,7.8). The instance in Figure 7.7 is the only one of the available ones where the feature `laufzeit` looses completely the effect on the output, in pursuit of the variable `moral` with category 1 (critical account other credits) that becomes the second feature with highest impact on the output, lowering it. The rest of the contribution is distributed over many other variables, so that any of these variables has a really high effect, of which the most outstanding are the already mentioned `moral`, or other like `hoehe` ,`verw`, `famges` or `alter`.

When analyzing the graphics of SHAP the explanations mostly agree with LIME and among both variants of SHAP. The principal result is that the features that had an effect of pushing up the prediction value in LIME, in SHAP also have a contribution of rising the output or they do not contribute ($\phi_i = 0$), but they do not lower the output, i.e. the corresponding SHAP values are not negative. Furthermore, the most important variable for all explained instances in Figures 7.3 to 7.8 is `laufkont`. Nevertheless, SHAP mostly rests importance to `sparkont` and `laufzeit`, although they still stay between the most important ones, among the ones we find also now `moral`, `hoehe` ,`verw`, `famges` or `alter`.

So far just individual instances have been analyzed, getting local explanations. Nevertheless, SHAP offers the possibility of using all the local explanations of the test

set together so that, via combining the local explanations, it can be obtained a global explanation of the behavior of the predictions model, retaining local faithfulness [22]. In Figure 7.9 there are shown 4 graphics that correspond to TreeSHAP. Each graphic is a force plot, such as the ones of Figures 7.3 to 7.8, where pink means the feature rises the output value to 1 and blue, that the feature lowers it. Nevertheless, this time the force plot corresponds not to an individual instance, but the force plots of all instances in the test set are rotated 90 degrees and put together. In particular in Figure 7.9, instead of showing the effect of all the variables together, just four features have been selected, and even more, disaggregated into four different graphics. These features are `laufkont`, `moral`, `age` and `gastarb`, i.e. the one we identified in the individual plots in Figures 7.3 to 7.8 to be the most important, two of the importants, and one that has be considered to not having contribution effects. When oberving Figure 7.9, this is confirmed, and even more `laufkont` pushes up the value when it is class 4 or 3 and pushes it down when it is of class 1 or 2. In Figure 7.10 the same graphics are shown for KernelSHAP, and they turn out to be very similar to the ones of TreeSHAP.
Sorting the features by the sum of the SHAP values over the whole test set, the overview of the most important features for the model are obtained, as in Figure 7.1 where it can be seen the mean of the absolute SHAP value for each feature.

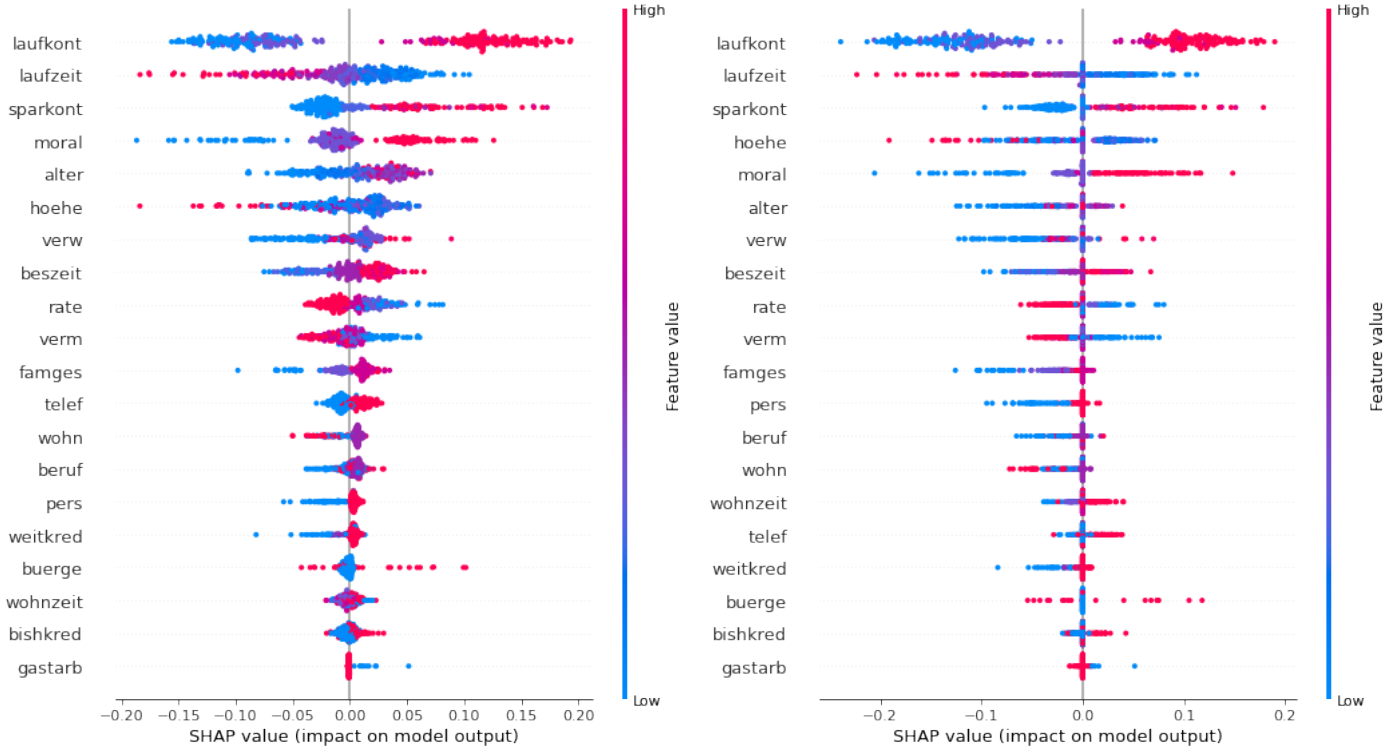


(a) TreeSHAP (b) KernelSHAP

Figure 7.1: Global view of important features according to SHAP

It follows from Figure 7.1 that as already expected the most important variable is `laufkont` followed by `laufzeit` and `sparkont`, and the non important ones are `gastarb` and `bishkred`. TreeSHAP and KernelSHAP give approximately the same order in the ranking, permuting at most three positions among the non important ones, or one position among the important ones.

At least let us take a view on Figure 7.2, where in addition to the ranking of the features, all the instances in the training set are represented for each feature with the corresponding SHAP value and colored depending on the value of the feature. For categorical features with many categories this gives no information, but for categorical features that have ordinal meaning or for quantitative features this is useful to determine which classes or values of the variable are responsible for positive or negative SHAP values. Therefore take a look on the most important variables `laufkont`, `laufzeit` and `sparkont`. When `laufkont` has a high value, i.e. the status of the account is high, the SHAP value is positive, while, for low values, i.e. low status of the account, the SHAP value is negative. For the quatitative feature `laufzeit`, the low values are the ones that increase the output value. And at least, for the variable `sparkont`, alike for `laufkont` , high values, i.e. having high savings, the SHAP values are positive, otherwise negative.



(a) TreeSHAP (b) KernelSHAP

Figure 7.2: Global view of important features according to SHAP

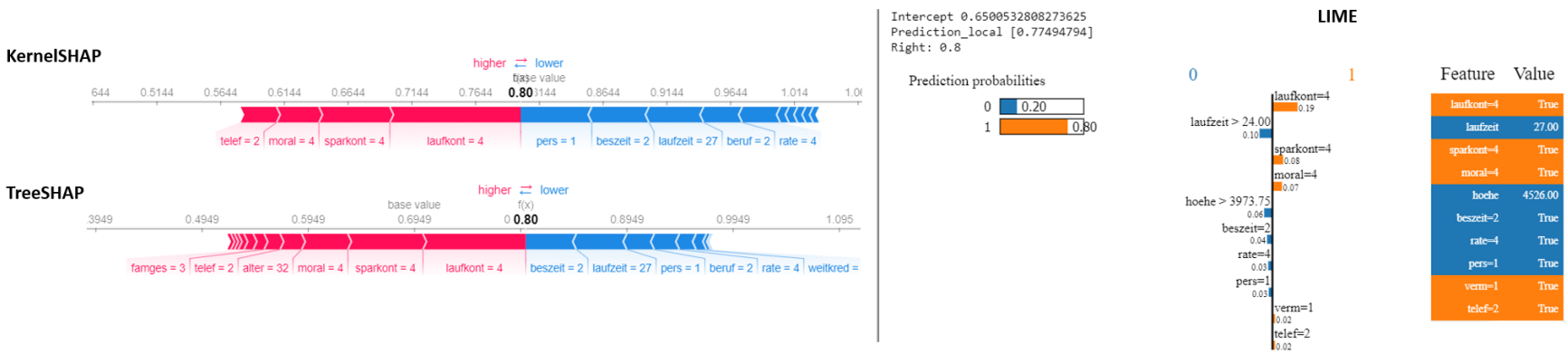


Figure 7.3: Instance number 95

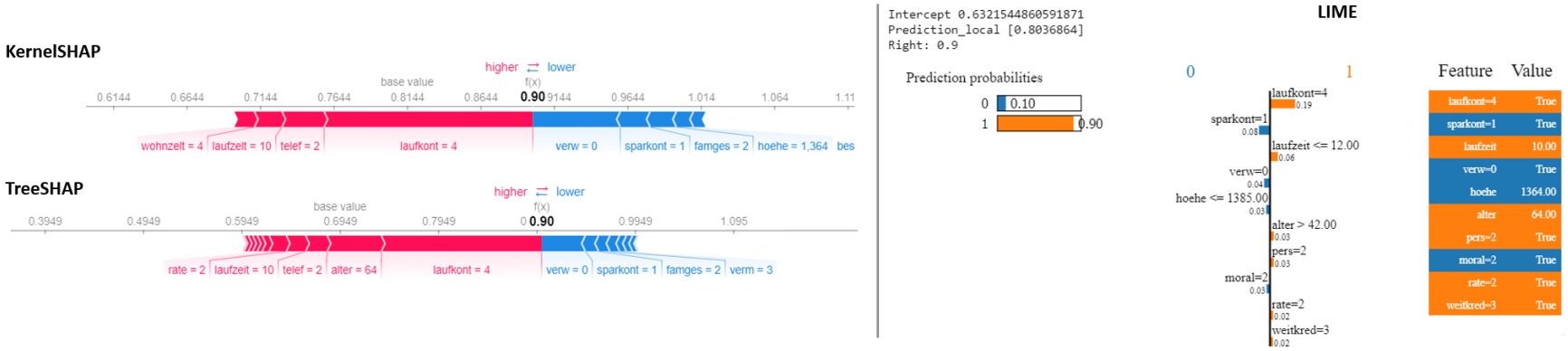


Figure 7.4: Instance number 134

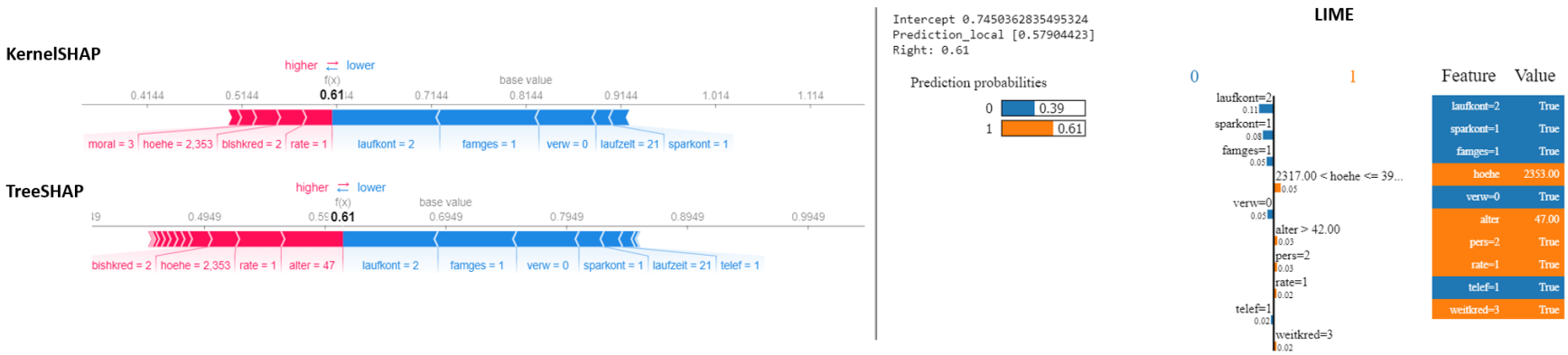


Figure 7.5: Instance number 283

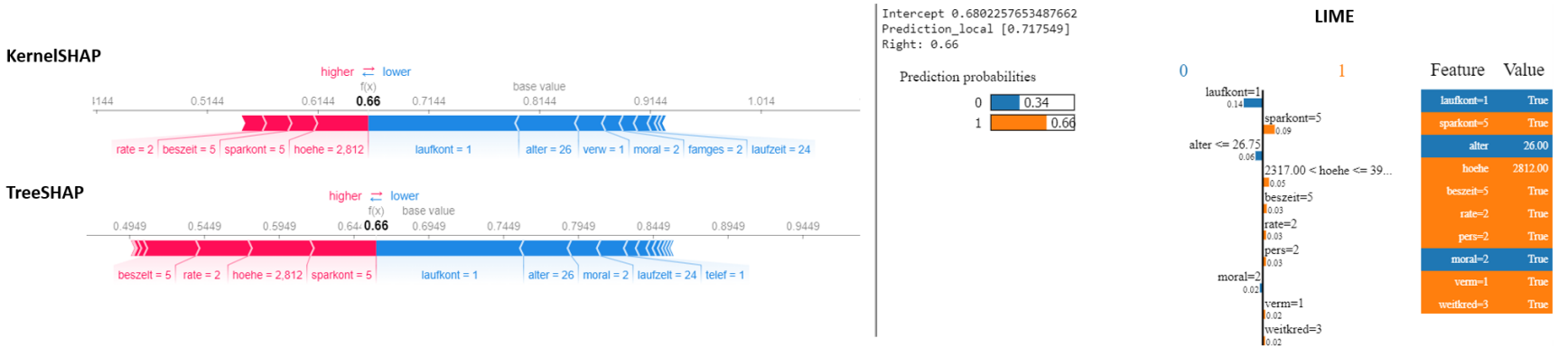


Figure 7.6: Instance number 389

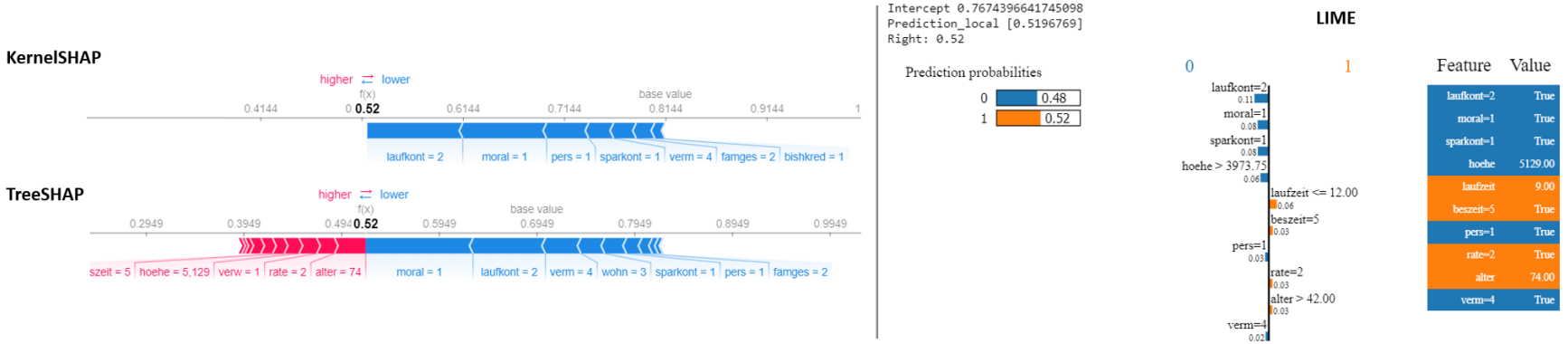


Figure 7.7: Instance number 814

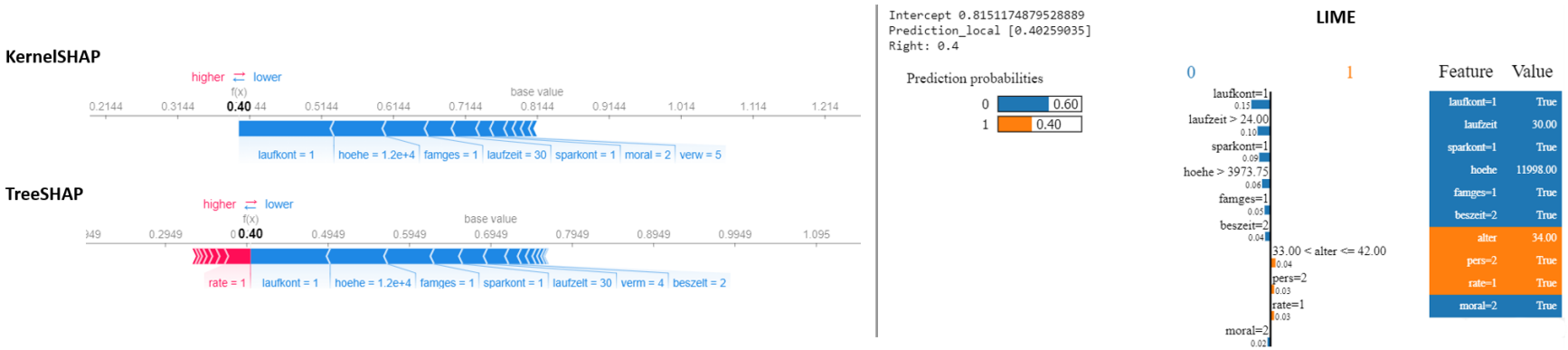


Figure 7.8: Instance number 975

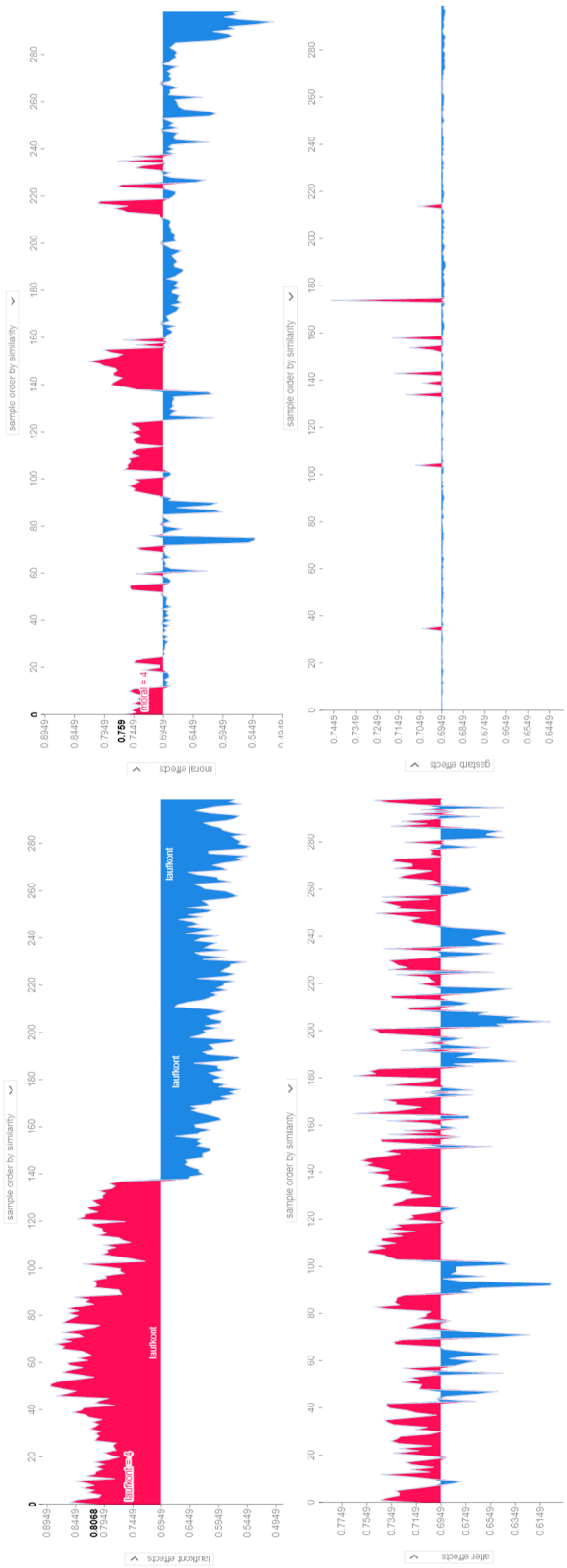


Figure 7.9: Force plot TreeSHAP

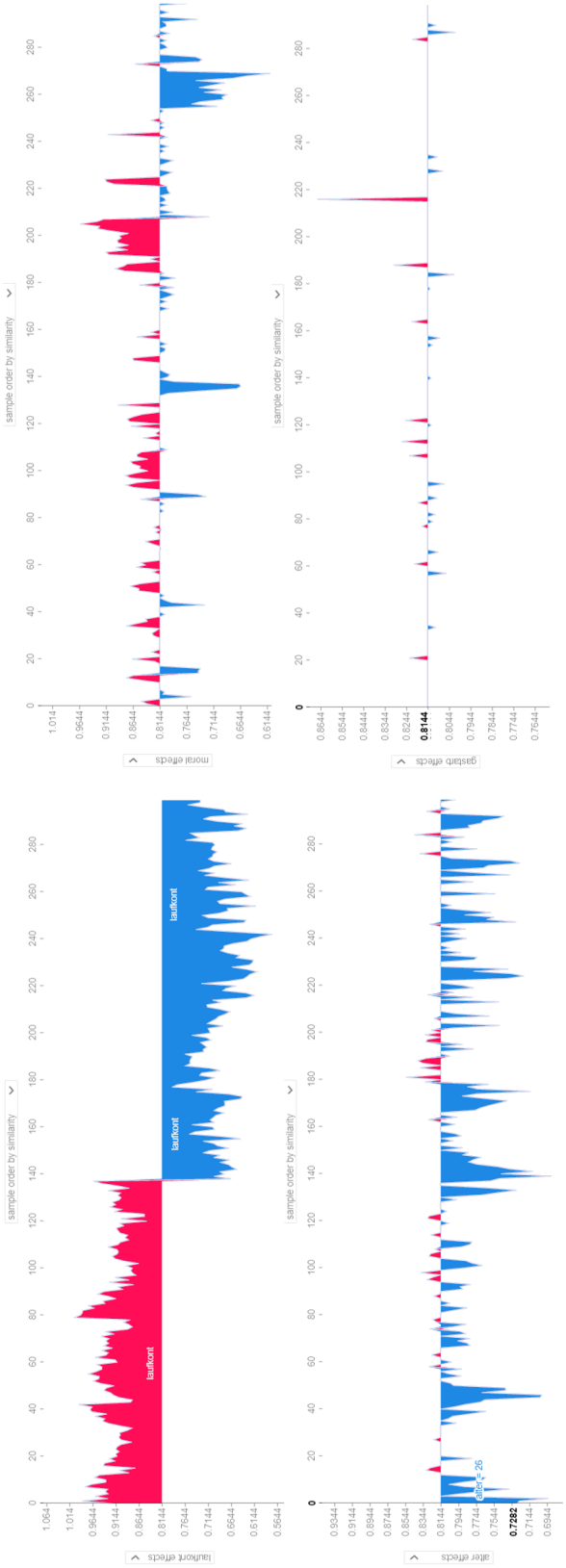


Figure 7.10: Force plot KernelSHAP

7.1.1. Conclusions of SouthGermanCredit

When analyzing the constructed random forest with LIME, KernelSHAP and TreeSHAP it stands out that all three methods conclude similar results for the local predictions of individual instances, giving the major effect to `laufkont`. Furthermore, all three criteria coincide in deciding whether the feature pushes down or up the prediction. In addition, SHAP provides a global view by putting all the local explanations of the instances in the test set, ranking `laufkont` as the most important variable for the model, followed by `laufzeit` and `sparkont`. High values of `sparkont` and `laufkont` mean high positive SHAP values, i.e. the predictions depend mostly on the saved money and the money in the account.

7.2. COMPAS

The second data set is called `COMPAS` (Correctional Offender Management Profiling for Alternative Sanctions). This dataset is a collection of different features for 7214 criminal defendants in Broward County, Florida, that aim to predict the recidivism. This data set stems from the use of an algorithm called COMPAS in Wisconsin that is used, as a black-box, to predict whether the criminal will or will not re-offend. After sentencing 2016 an individual for high risk of recidivism, this algorithm generated high controversies [10]. Therefore, the authors of [19] collected the data set which will be used in this section, where it must be noticed that one of the variables represents if the individual really re-offended in the following two years, and another of the collected variables represents what the COMPAS algorithm predicted. They analyzed, as well as in [34], whether the COMPAS algorithm predicts in an "unfair" way due to racial, age or sex based aspects.

In this section, it will be analyzed, on the one hand, an explanation according to LIME and SHAP of a random forest trained with the real variable of recidivism as response variable and, on the other hand, an explanation also according to LIME and SHAP of a random forest, but this time trained with the COMPAS prediction as response variable. The explicative variables that will be used from the given data set are represented in Table 7.2.

variable	**Classes (when categorical)**
c_charge_degree	0 : Felony
	1 : Misdemeanor
race	0 : African-American
	1 : Asian
	2 : Caucassian
	3 : Hispanic
	4 : Native-American
	5 : Other
age_cat	0 : between 25 and 45
	1 : over 45
	2 : under 25
sex	0 : Female
	1 : Male
priors_count	—

Table 7.2: Variables in COMPAS

7.2.1. Real recidivism

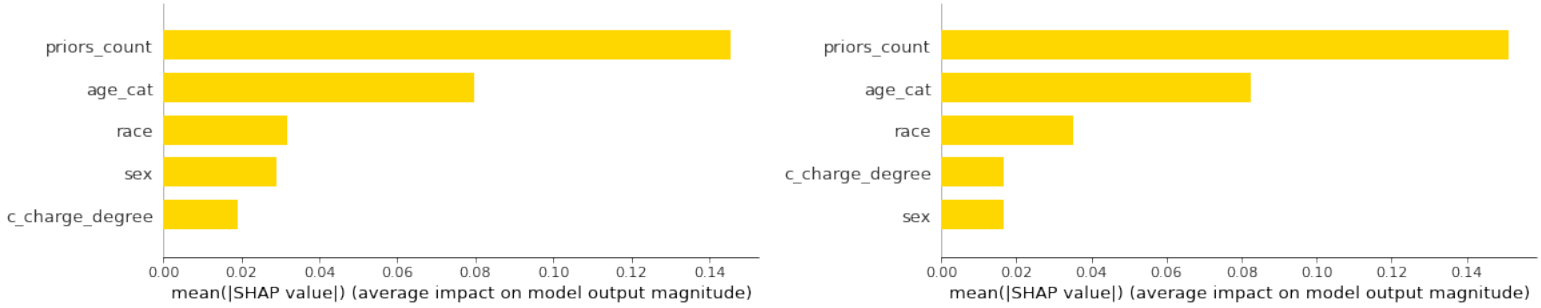
As already mentioned, the first analysis will be done using as response variable the variable of real recidivism (`is_recid`) in the following two years. This variable is binary with classes 0 (did not re-offend) and 1 (did re-offend).

As well as was done with the `Credit`, we will first take a look at predictions for individual instances via KernelSHAP, TreeSHAP and LIME and afterwards, induce a global overview with SHAP. Notice that here $f(x)$ also gives the probability of x to correspond to class 1, and consequently prediction values (probability) next to 1 mean that the instance x is predicted to re-offend (class 1) and next to 0 mean that the instance x is predicted to not re-offend. (class 0).
The Figures 7.14 to 7.19 show the explanations of the predictions of the random forest for six individual instances, where the first three instances were predicted by the random forest to not re-offend (class 0) and the three last ones to re-offend (class 1). Let us first go in deep into LIME explanations. For all six instances, the feature `priors_count` is selected as the one that induces most effect on the prediction. For the first three (Figures 7.14,7.15 and 7.16) the feature takes low values, and the feature contributes to lower the output, i.e. to take the prediction to class 0. For the last three (Figures 7.17,7.18 and 7.19) , the number of `priors_count` is high, over 5, and therefore this feature contributes rising the prediction value in the direction of 1. The next two features that effect most the output are `sex` and `age_cat`. In case the feature `sex` takes the category 1, i.e. male, it can be observed that the effect is pushing up the output to re-offend (Figures 7.14,7.15,7.17 and 7.18), while when it takes the category 0, i.e. female, the effect is pushing down the output to no re-offend (Figure 7.16). For the feature `age_cat`, when it takes the value 0 or 1, i.e ages between 25 and 45 or age over 45, respectively, the effect is to push down in direction to class no re-offend, the output (Figures 7.14,7.15,7.17,7.18 and 7.19), while `age_cat`=2, i.e. age under 25, pushes up the output in direction to class 1 (Figure 7.16). The variables `race` and `c_charge_degree` are the variables that contribute less to the predictions, although they also have some effect. When `race` takes the value 0, i.e. African-American, the feature pushes the output to class re-offend (Figures 7.16 and 7.17), while all the other classes push it down. For `c_charge_degree`, when the class is 0, i.e. felony, it pushes up the prediction, while class 1, i.e. misdemeanor, pushes it down. Nevertheless, it is worth to emphasize that the contribution of this two features is very low, especially compared to `priors_count` contribution (the first have always values ϕ_i around 0.01, while the second has values around 0.13 or even $.33$.

When doing the explanations with SHAP the analysis results as follows. For the instances of Figures 7.15 to 7.19, the variable `priors_count` is identified as a variable with high effect, as well as identified LIME. In fact, for the mentioned instances except for the one in Figure 7.16, where `priors_count` shares the first place in importance with `age_cat`, and in 7.15 in the TreeSHAP version, where

many variables are equaled in effect, the feature `priors_count` is considered the most important. Furthermore, the variable `age_cat` stays as in LIME as one of the features that follow `priors_count` in importance in most of the Figures. Nevertheless, together with `age_cat`, LIME positioned in most of the given instances `sex` mostly as important as `age_cat`, but SHAP (Kernel and Tree) lower its effect, and gives the variable `race` a higher effect, specially in the instance of Figure 7.14, where it is adjudicated the highest effect (remember this is the instance we left out in the beginning when saying that SHAP considered in most of the given instances `priors_count` as the feature with most effect). The variable `c_charge_degree` is also considered by both SHAP variants as the one that influences less the prediction of the given instances. It is worth mentioning that, while in LIME some patterns were observed relating the class of the feature with the direction in which it pushes the output, in SHAP, for the given instances, it cannot be identified such patterns and therefore it will be needed to take a global overview such of the one done in the data set `SouthGermanCredit`. Nevertheless, what can be concluded is that, locally, LIME and SHAP differ for some instances when speaking about the second to last most influent features, but both sentence mostly the same feature as most important. In addition, the local results of KernelSHAP and TreeSHAP do not agree at all, as happened with `SouthGermanCredit`, but they mostly agree, indeed for all given instances except for the one of Figure 7.15.

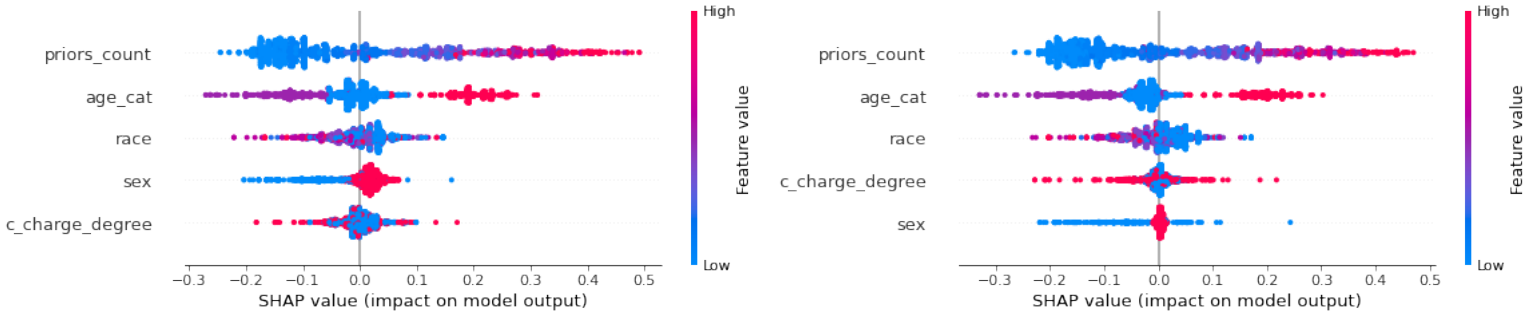
Let us now take a global view putting all the results of SHAP for the instances in the test set together. When ranking the features like in Figure 7.11, according to the mean of the absolute SHAP values over the whole test set, the feature placed with the highest SHAP value is, as expected, `priors_count`. With a SHAP value wide below, the feature that follows it is `age_cat`, and also wide below, the next feature is `race`. The last two features are `sex` and `c_charge_degree` according to TreeSHAP, and the other way around according to KernelSHAP. Nevertheless, both last features have, on average, approximately the same SHAP values.



(a) TreeSHAP

(b) KernelSHAP

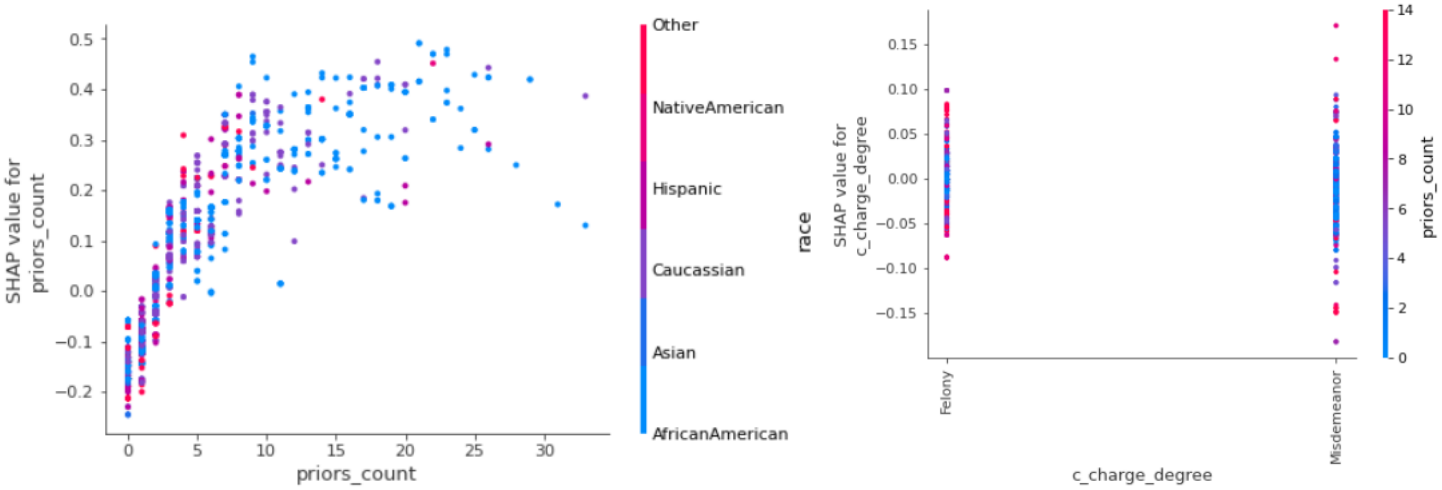
Figure 7.11: Global view of important features according to SHAP



(a) TreeSHAP (b) KernelSHAP

Figure 7.12: Global view of important features according to SHAP

When representing the SHAP values for each feature for every instance in the test set, colored by the feature value, we get the Figure 7.13, where the results obtained are very similar for Tree and KernelSHAP: High values of `priors_count` imply positive SHAP values, where the instances are distributed between SHAP value=0.2 and 0.5, i.e., the effect of this feature taking high values is mostly the highest, and taking low values imply negative SHAP values, but the instances are grouped around SHAP value=-0.15, what means that the effect is not that high as for high values. This captures what we have already seen in Figures 7.18 to 7.19: SHAP not always states `priors_count` as the most important one. In order to get a deeper understanding of this feature let us take a look at Figure 7.13(a), where it can be observed the SHAP value of this variable for the different values of the feature, colored by the variable `race`. As it can be seen, in addition to what already explained, most of the instances of high `priors_count` correspond to `race`=African-American, and therefore the SHAP-value of African-American at `priors_count` is higher than for other races. Nevertheless, going back to Figure 7.13, for the variable `race` there are no clear patterns, but it is worth mentioning that the blue values are grouped around the zero on the positive side of the SHAP values, i.e. when the feature `race` has a low value, and remember the lowest class of `race` is 0=African American, the SHAP value of `race` is zero or positive but very low, i.e. the random forest does, according to SHAP, not attribute high SHAP values, and therefore not effects pushing the prediction to class of re-offend, to African American. Let us next analyze the behavior of the variable `age_cat` with categories: 0 (the age is between 25 and 45), 1 (age is over 45), and 2 (age is under 25). According to the graphic, the blue points correspond to class 0, the lilacs to class 1 and the red ones to class 2. It follows that when the individuals are under 25 the corresponding SHAP value is positive, i.e. the young is hold accountable for recidivism; when they are between 25 and 45 the SHAP value is next to zero or negative, but in absolute terms, very small; and when they are over 45, the SHAP value is negative, i.e. the maturity pushes the prediction into the direction of no recidivism. For the variable `sex` it is worth mentioning that, when it takes the class female, the



(a) Dependence features `priors_counts` and `race`

(b) Dependence features `c_charge_degree` and `priors_counts`

Figure 7.13: Dependencies of features according to TreeSHAP

corresponding SHAP value is negative or a low positive value, while when it takes the class male, the value is around zero or a low positive value, i.e. in general the category female is responsible for pushing down the output value or it is not attributed effect, while the category male is responsible for pushing thinly the output value to the class re-offend, or it is not attributed effects. At least, the feature `c_charge_degree` is only attributed effects when it takes the class misdemeanor. Nevertheless, depending of the instance it pushes the output up or down. Figure 7.13(b), shows the the relation of `c_charge_degree` with the feature `priors_count` and stays that the higher SHAP values for `c_charge_degree`=misdemeanor correspond to instances where the number of `priors_count` are very high, while the low ones, correspond to instances where the number of `priors_count` are intermediate.

Figures 7.21 and 7.20 display the force plots for all instances in the test set, for KernelSHAP and TreeSHAP, respectively, disaggregated by features, restating what is explained above, variable by variable.

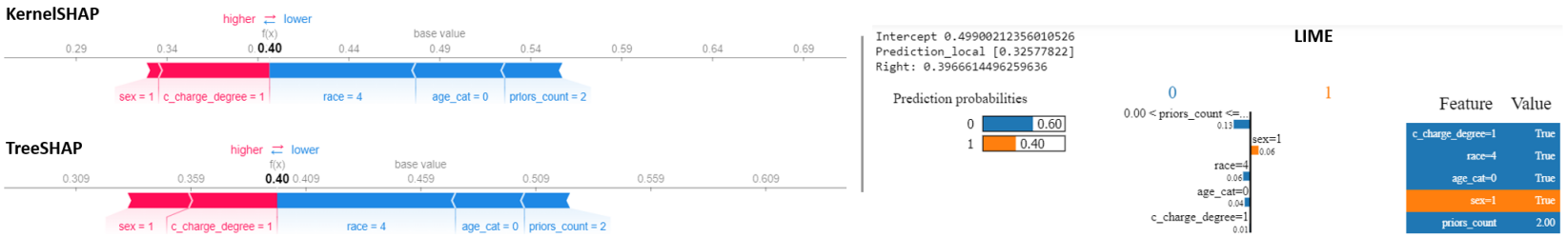


Figure 7.14: Instance number 4611

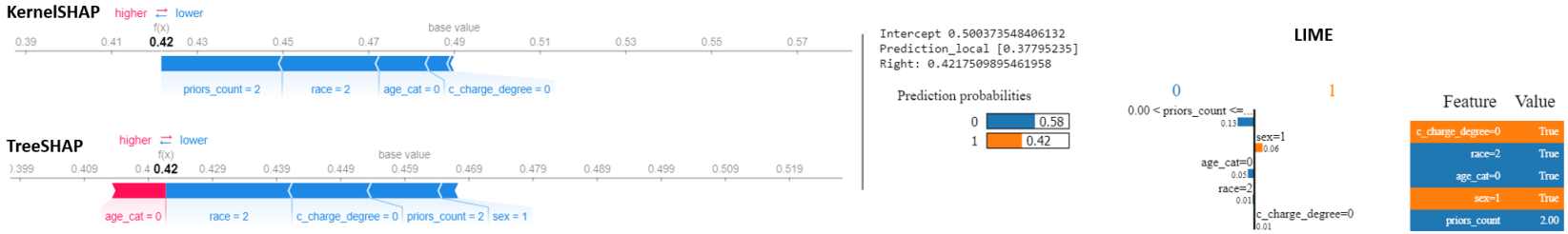


Figure 7.15: Instance number 5277

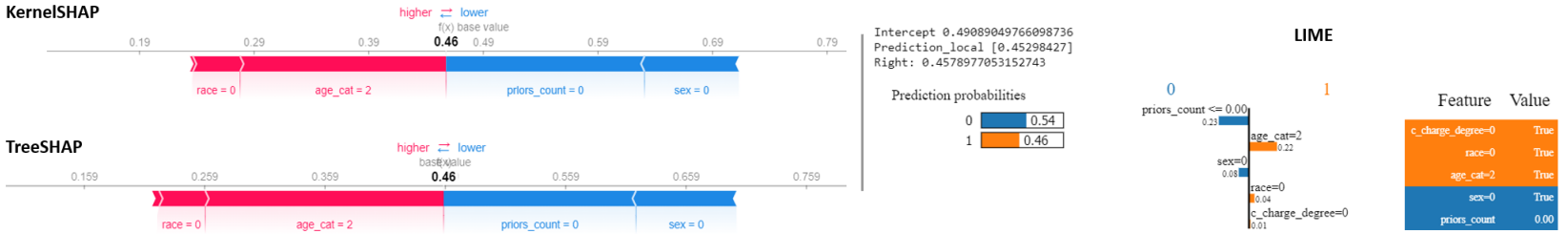


Figure 7.16: Instance number 2962

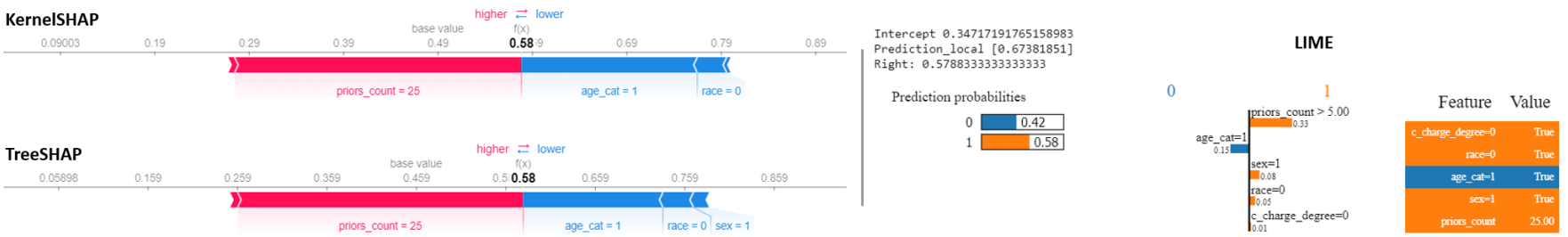


Figure 7.17: Instance number 833

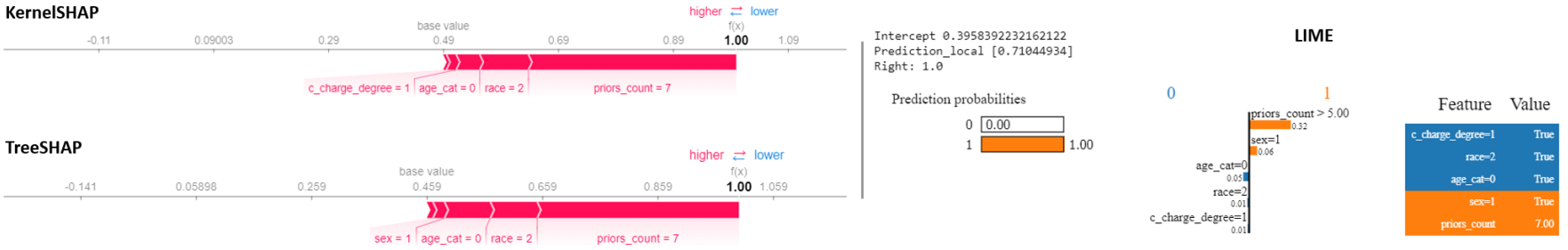


Figure 7.18: Instance number 2071

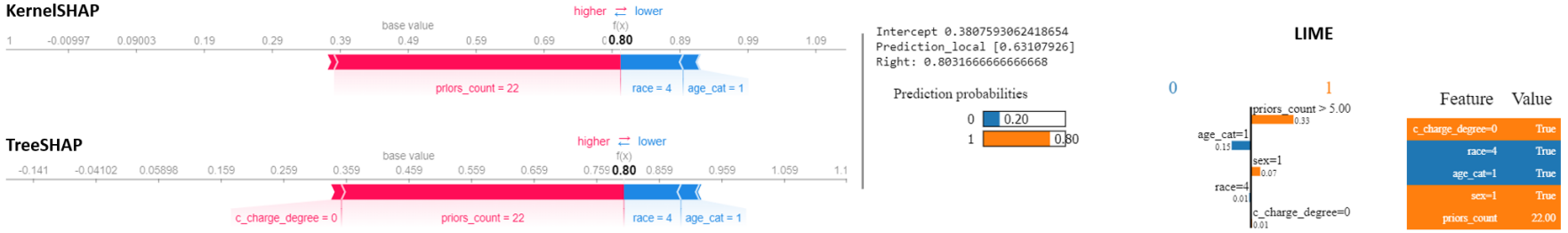


Figure 7.19: Instance number 6143

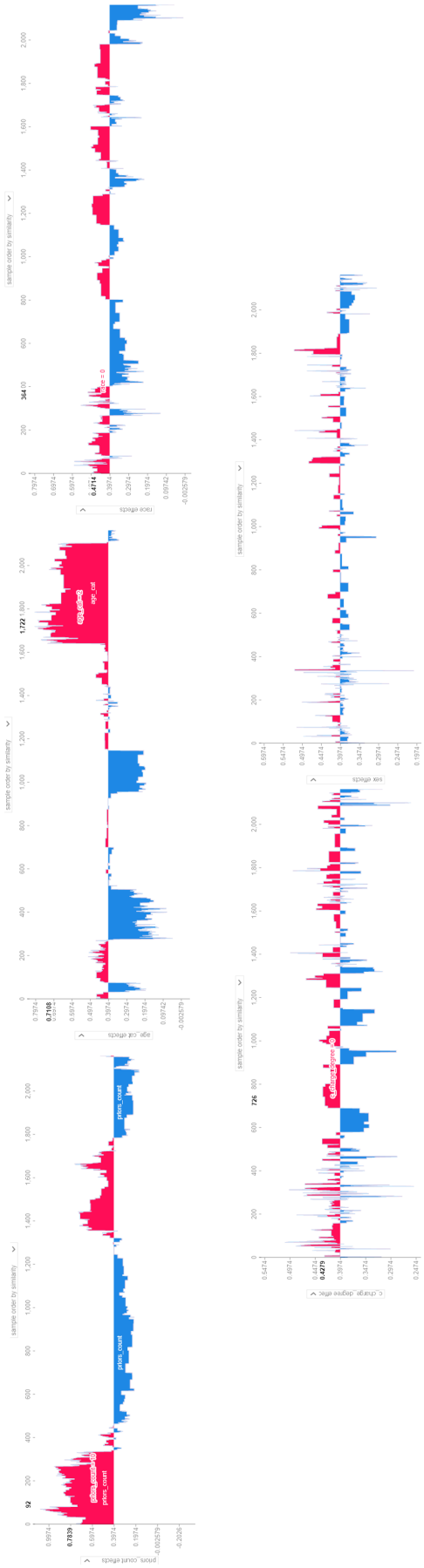


Figure 7.20: Force plot TreeSHAP

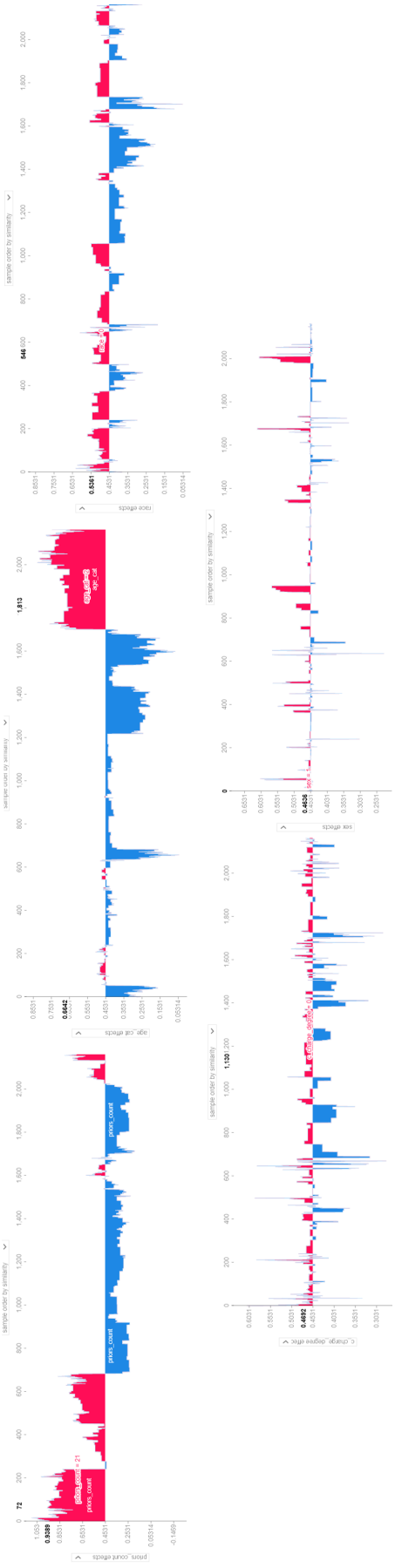


Figure 7.21: Force plot KernelSHAP

7.2.2. COMPAS recidivism

The second analysis done with the COMPAS data set consists in constructing the random forest using this time, as response variable, the variable that expresses the prediction of COMPAS algorithm of re-offending the criminals. Indeed, the predictions done with COMPAS are leveled as : *low, medium* and *high* risk of recidivism. Nevertheless, in order to make possible a comparison with Section 7.2.1, the level *low* has been set as no risk of recidivism (class 0) and classes *medium* and *high* are set as risk of recidivism (class 1).

LIME results in Figures 7.23 to 7.28 indicate that, as well as in Section 7.2.1, the feature `priors_count` is given, in most of the Figures the highest (in absolute terms) SHAP value (just in Figure 7.25 it is ranked as the second one with a SHAP value =0.28, very close to the first one that has SHAP value =0.30) , and therefore, identifying it as the feature with most effect of rising, when its value is over 5 (Figures 7.26,7.27,7.28) or lowering in other case the output value (Figures 7.23,7.24, 7.25). Nevertheless, while in Section 7.2.1, the next two most important features were on most of the local predictions `age_cat` and `sex`, when using COMPAS predictions as response variables, `age_cat` is still one of them but instead of `sex` we have `race`. This time also just `age_cat`=2, i.e. age under 25 and `race`=0, i.e African-American, rise the output value in direction of class 1, re-offend. The features `c_charge_degree` and `sex` are attributed low values, in absolute terms, and do not effect much the output. Moreover, the above discussion can be mostly transposed to the results obtained by KernelSHAP and TreeSHAP, where the ranking for each instance is very similar to the one of LIME and, except for the instance of Figure 7.27, where the methods differ a bit, the features effect in the same direction as exposed by LIME.

Moreover, when observing the global view given by TreeSHAP (the one given by Kernel is not exposed because it is very similar to the one of TreeSHAP) of Figure 7.22, the discussion above is reaffirmed globally. The most important feature is `priors_count` followed by `age_cat` and `race`. Additionally it can be deduced that when `priors_count` is high, the corresponding SHAP value is also high, and correspondingly when it is low, as already happened in Section 7.2.1. As well as happened in Section 7.2.1, for the feature `age_cat`, the SHAP values of this variable is positive when `age_cat`=2, i.e. age under 25, mostly zero when `age_cat`=0, i.e age between 25 and 45, and negative when `age_cat`=1, i.e. age over 45. Nevertheless is worth mentioning that, this time, there is no overlapping, and all three zones are clearly separated. For the feature `race`, it can be observed that all races effect with a negative SHAP value, except for `race`=0, i.e. African-American, that produce positive SHAP values contributing to class of re-offend. It is worth mentioning that while in Section 7.2.1, the blue values were grouped around the zero on the positive side of the SHAP values, but with low values, in Figure 7.22, most of these instances are

grouped around the SHAP value=0.1, and although it is not a high value, it is higher than the values attributed in Section 7.22, evidencing that `race`=African-American pushes the prediction to the class of re-offending, specially comparing to all the other races that obtained, not even SHAP value=0, but low negative SHAP values. For the feature `c_charge_degree`, the instances with class misdemeanor have a low (in absolute terms) negative SHAP value, while the ones with class felony have a low positive SHAP value, while in Section 7.2.1 felony was no attributed effect and for misdemeanor there were no clear patterns. At least, for the feature `sex`, the males are grouped around the zero, while the females are on both sides.

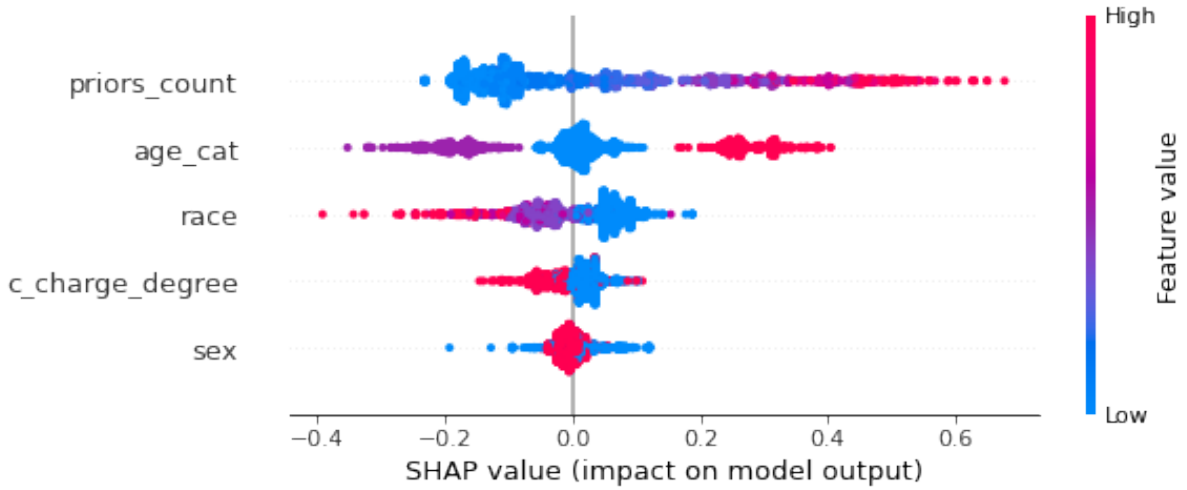


Figure 7.22: Global view TreeSHAP COMPAS

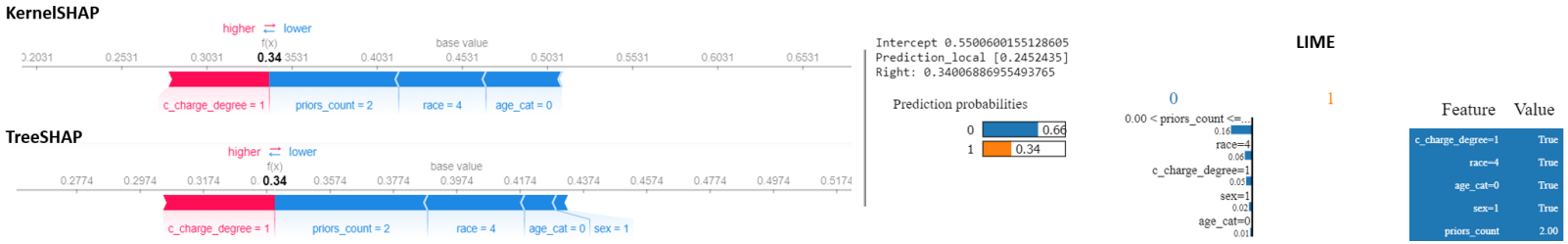


Figure 7.23: Instance number 4611

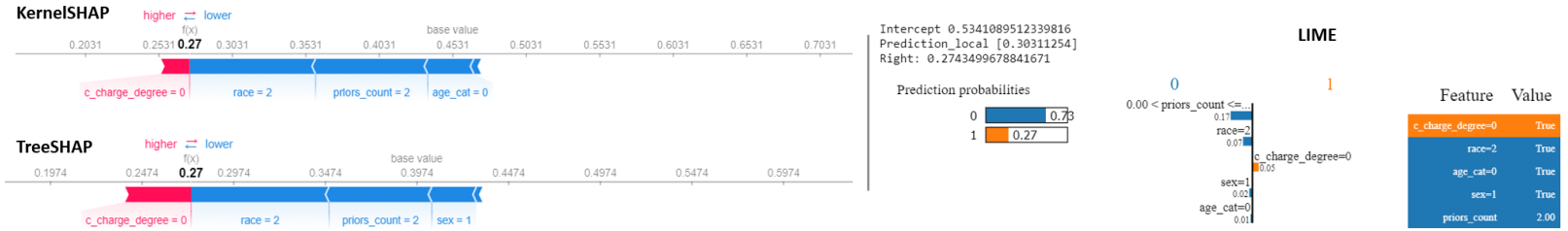


Figure 7.24: Instance number 5277

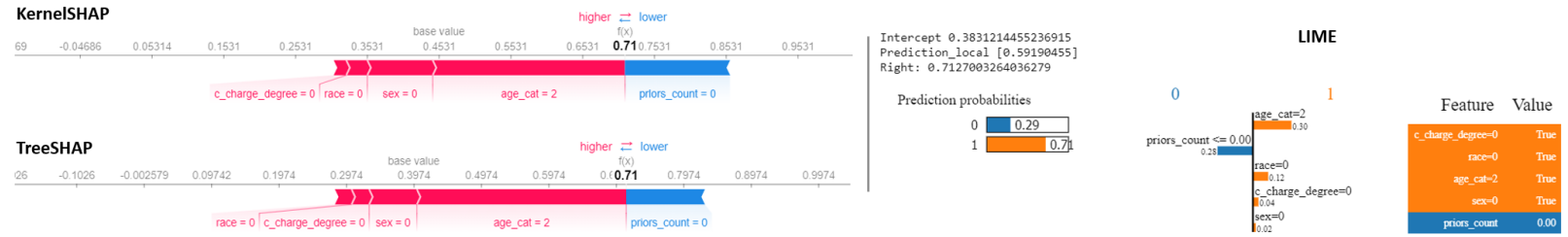


Figure 7.25: Instance number 2962

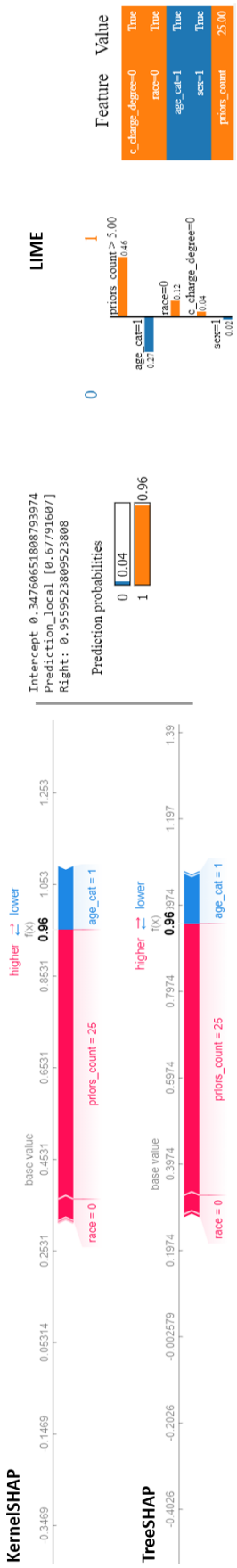


Figure 7.26: Instance number 833

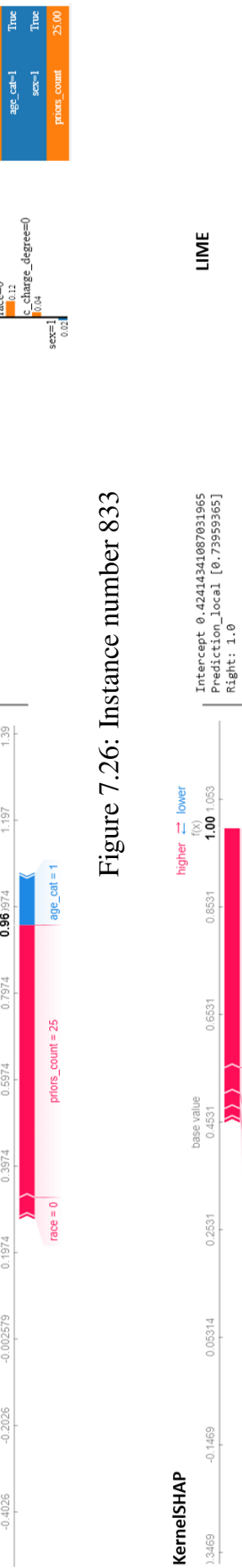


Figure 7.27: Instance number 2071

Figure 7.28: Instance number 6143

7.2.3. Conclusions of COMPAS

Both constructed random forests, one with the response variable of real recidivism, and the other with the response variable of COMPAS algorithm, do their prediction based mostly on `priors_count`. Nevertheless, not the whole effect is attributed to this variable, and the variable `c_charge_degree` is not attributed much effect, by any of the two models. The remaining variables are `sex`, `age_cat` and `race` but a fair model should never predict based on sex, age or race aspects. Both models attribute "globally" lower effects to this variable than to `priors_count` but not always zero. In both cases the ranking in variable importance is after `priors_count`, `age_cat`, `race`, `c_charge_degree` and `sex` (except for TreeSHAP in the random forest with the real response values that exchange `c_charge_degree` with `sex`). For the variable `age_cat` the pattern is clear in both cases: persons under 25 will re-offend, persons between 25 and 45 are not attributed any effects and being a person over 45 is a mitigating in terms of predicting recidivism. Nevertheless, although the variable race is labeled in both cases as the third most influential, the model with the real recidivism attributes zero or low effects aggravating the possibility of recidivism when being African-American, while the other races lower this probability. The second model, with the COMPAS algorithm variable, sets higher effects aggravating the possibility of recidivism when being recidivism than the first model, and considers an attenuating being of any other race. For both models the `sex` is not very influential, although the one with the real values consider that males have a higher probability of re-offending than females. Although `c_charge_degree` is not given much importance, the model with the COMPAS algorithm values identifies having committed a felony as an aggravating for recidivism, while the first model does not identify any clear patterns.

As can be seen in the above discussion, explaining prediction models like the random forests used for the COMPAS data set has an important role in fairness. It has exposed that the forest, especially the second one, attribute effects to age and race. Therefore it is important to know the explanations given by methods like LIME or SHAP, in order to be able to determine if the prediction is reasonable, or whether the assumptions of the model are not fair or right. It follows also that the features used by the model should be different. For instance, instead of race, it should be studied a new model using other variables like, incomes, job, etc.

7.3. Functional data

In this section, it will be analyzed the explanations of two random forests, each for a different data set of functional data, called `growth` and `tecator`.

As explained in Chapter 1, for using random forests we need a sample $\mathcal{D}$ of M observations, where for each observation a pair $(\mathbf{X}^j, Y^j)$ is associated, where $\mathbf{X}^j$ is a vector of $|N|$ variables. Nevertheless, the scenario when using functional data is, for each observation, a pair (X^j, Y^j), where X^j is a Riemann integrable function $X^j : [0, L] \to \mathcal{R}$. In order to operate with this function, X^j must be discretized into $|N|$ points $(l_1, ..., l_{|N|})$ of the interval $[0, L]$ (for each observation j, the sequence $(l_1, ..., l_{|N|})$ used must be the same one). After the discretization the new scenario is the initially desired where for each observation, there is a pair $(\mathbf{X}^j, Y^j)$, adding that the variables correspond with concrete instant of time (or length or height etc. depending of the units of the interval).

7.3.1. Growth

The first data set of functional data under study is `growth`. The data set consists on 93 individual, 39 of which are boys and 54 girls, for which the height from the age 1 to 18 has been collected, i.e. the height is the function X over the interval $[1, 18]$. Indeed, the function of the height has been discretized into 31 ages (not equidistant). The `growth` data set has been used first to train (70% of the set) a random forest that classifies the individuals into boy (1) or girl (0), and then to explain the predictions that the forest does to the instances of the test set (30% of the set) using TreeSHAP and LIME using as variables the discretization of three different functions:

- Original function
- First derivative
- Second derivative

Additionally the global view given by SHAP has been compared to the results of MDI.

Figures 7.32 to 7.35 show the explanation of individual instances by SHAP and LIME when using the original function, the first derivative or the second derivative as explanatory variables. Figures 7.32 and 7.33 correspond to girls of the test set, and Figures 7.34 and 7.35 to boys. As it can be seen, except for when using the second derivative in Figure 7.34(c), the random forests classify correctly each individual. Let us now see the explanations of such predictions according to LIME. For the original function, for any instance, the most important ages are the ones between 16 and 18, although ages between 11 and 13 do also influence. It can be observed that for the given

Age	**11**	**11.5**	**12**	**12.5**	**13**	**16**	**16.5**	**17**	**17.5**	**18**
q_1	131.6	133.5	135.6	137.6	139.5	152.3	152.6	153.2	153.5	153.6
q_2	143.7	146.1	148.6	151.7	155.0	164.8	165.1	165.4	165.5	165.6
q_3	147.7	150.5	153.8	157.0	158.9	169.8	170.5	171.1	171.8	172.5
q_4	151.3	154.5	158.1	161.5	165.3	176.1	178.1	178.5	178.5	178.7
q_5	170.2	173.5	176.0	177.3	178.3	190.7	191.3	191.7	192.2	192.8

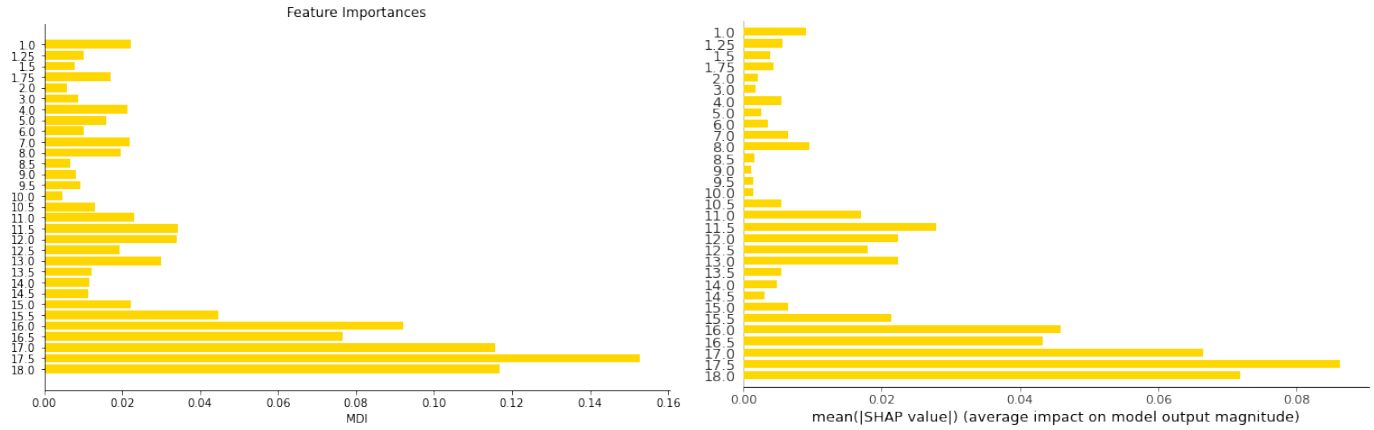
Table 7.3: Quartiles of most important ages for instances in Figures 7.32 to 7.35 according to notation used in Section 5.2

Figures, for ages 11 and 13, when the height takes a value between q_3 and q_4 or between q_4 and q_5 the contribution of the features is lowering the output value in direction to class girl (0), while for ages between 16 and 18, when the height is between q_3 and q_4 or between q_4 and q_5 the contribution of the features is rising the output value in direction to class boy (1), and the responsible for lowering the output are the heights under q_3. The values of $q_k, k = 1, .., 5$ for the most important ages are given in Table 7.3. Additionally it can be followed that for the local instances of the referred Figures, the ages in the same ranges have similar influence. SHAP results are very similar. Furthermore, same ages mostly influence the output into the same direction in both methods.

Using the first derivative of the function instead of the function itself, the results given by LIME and SHAP are mostly equal, identifying ages approximately between 13 and 16 as the responsible for pushing down or up the final output, i.e. although the height is different for girls and boy at the ages of 16 to 18 and 11 and 13, so that these ages are the most significant for classifying according to the age, the change in the height is different for girls and boys in the ages between 13 and 16. For the second derivative approximately the ages between 11 and 16 are identified for every mentioned instance as significant by SHAP and LIME.

When taking a look at the global view given by SHAP, the above exposed results for the given instances can be extrapolated to the whole test set. Figures 7.29(b), 7.30(b) and 7.31(b) show the mean of the SHAP values for each variable over the whole test set. There can be identified different mountains and valleys, where the mountains correspond approximately with the ranges as identified as important above. In fact, for the original function (Figure 7.29(b)) the features identified as important are ages between 11 and 13 and between 15.5 and 18.

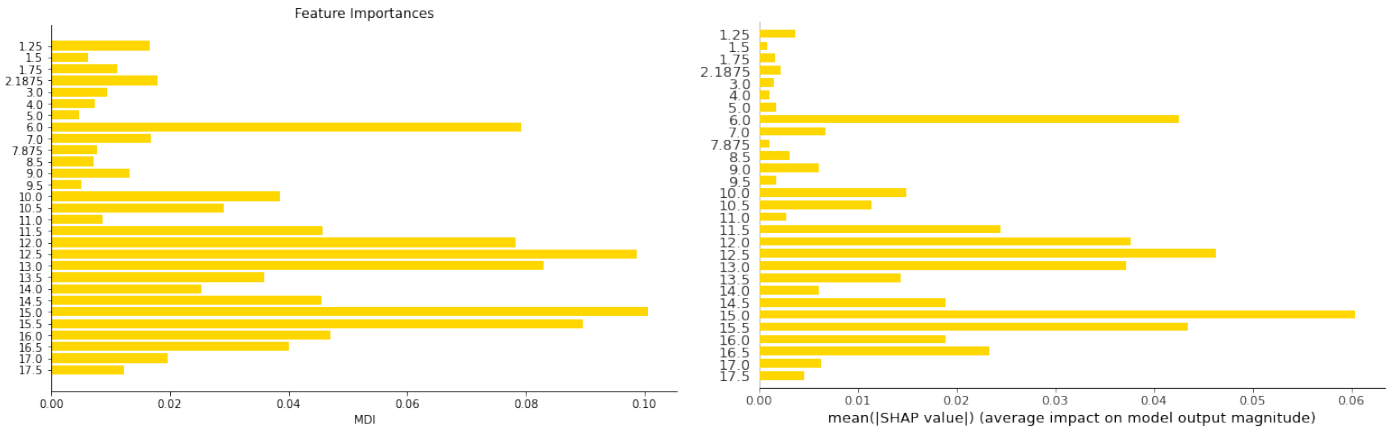
For the first derivative (Figure 7.30(b)) there is just one mountain between ages 12.25 to 16.25. At least, the second derivative (Figure 7.31(b)) show that the important ages are the ones between 12.75 and 16.25. Consequently, it can be deduced that for this data set of functional data the ages identified as important by the global view



(a) MDI (b) SHAP

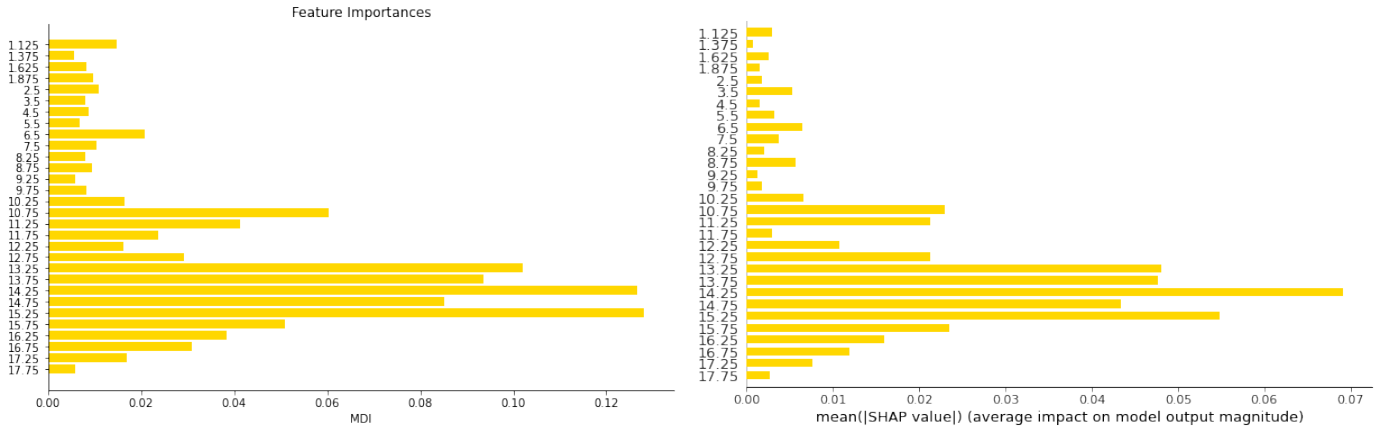
Figure 7.29: Global view of important features according to MDI and SHAP

of SHAP are grouped in ranges, so that, similar ages have the same effect on the prediction.



(a) MDI (b) SHAP

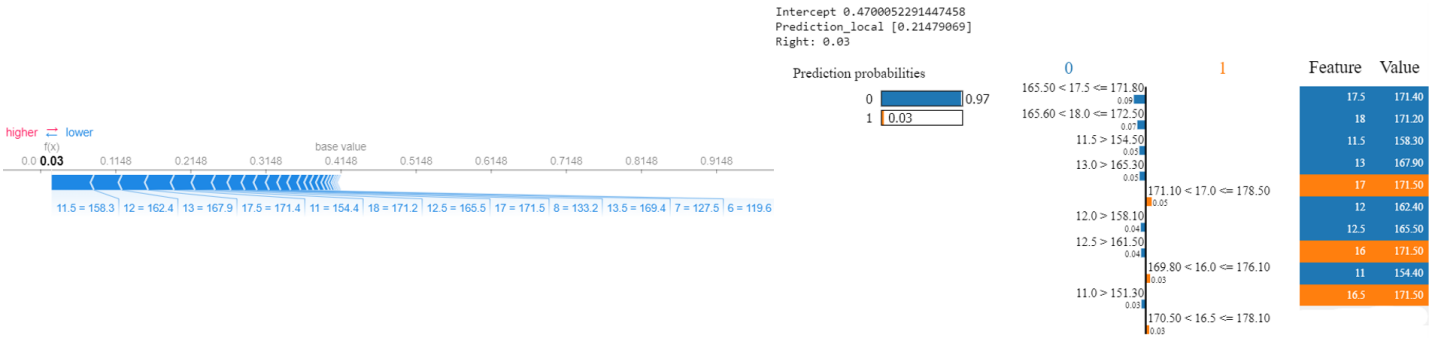
Figure 7.30: Global view of important features according to MDI and SHAP: first derivative



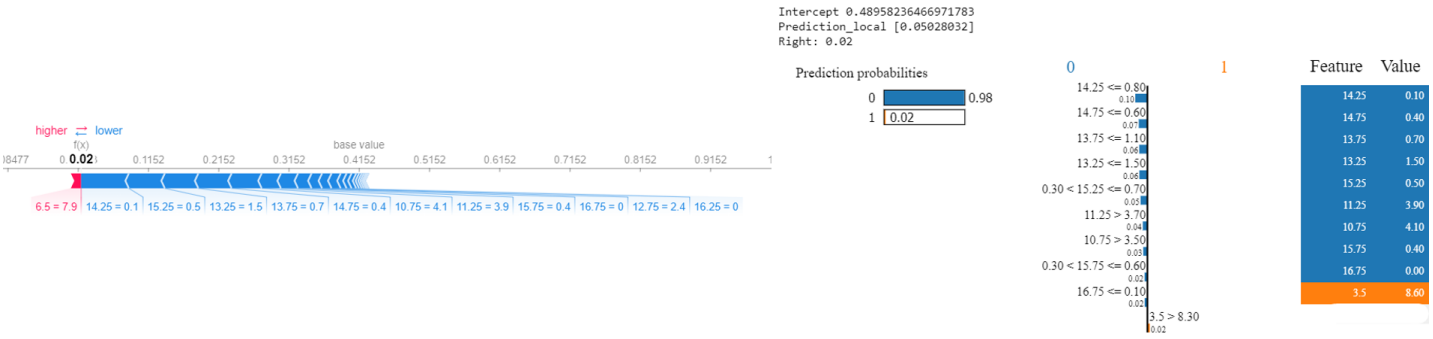
(a) MDI (b) SHAP

Figure 7.31: Global view of important features according to MDI and SHAP: second derivative

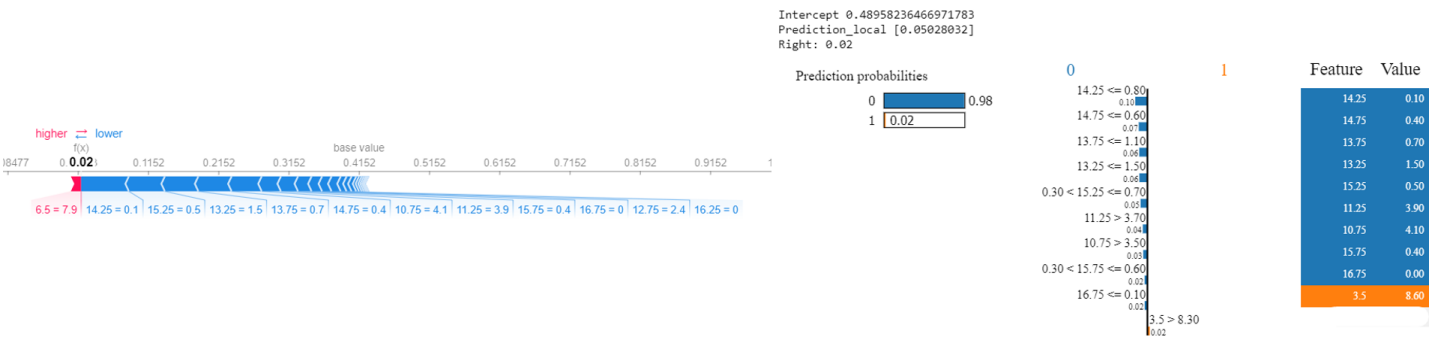
Moreover, in Figures 7.29 to 7.31 show in (a) the MDI values for the different ages in all three cases: original function, first derivative and second derivative. It can be observed that the results are very similar to the ones obtained by the global view of SHAP in Figures 7.29 to 7.31(b), because in all three cases the same age-ranges are identified as (un)important. Nevertheless, MDI just gives a global vision of the model while SHAP is able also to submit local explanations.



(a) Original function

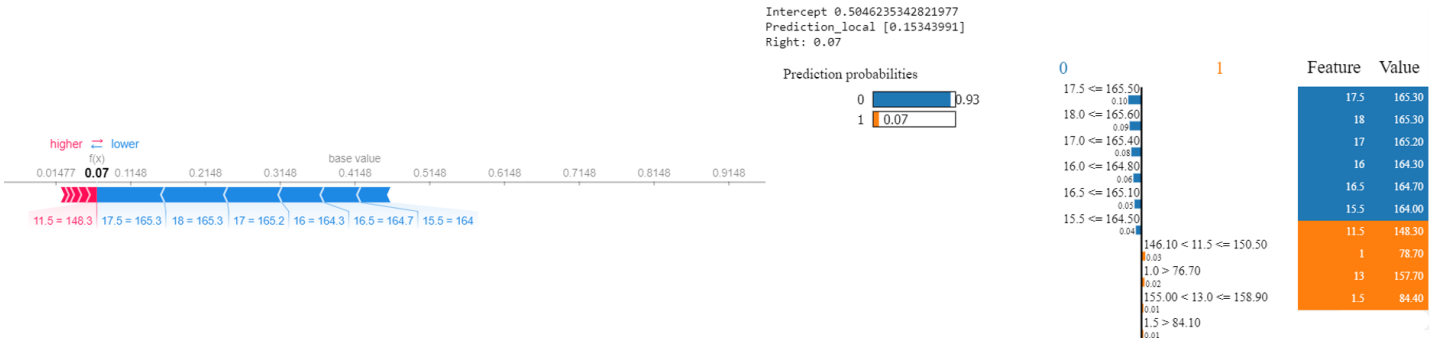


(b) First derivative

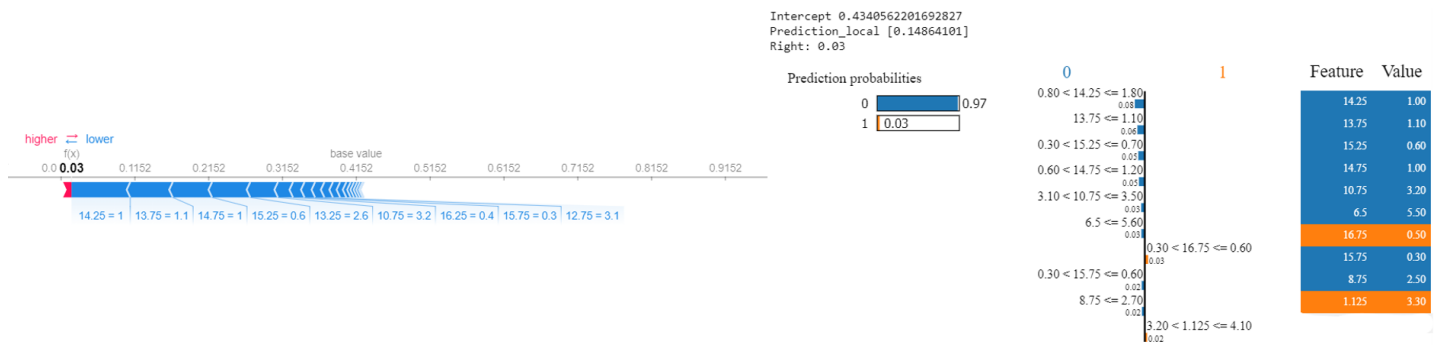


(c) Second derivative

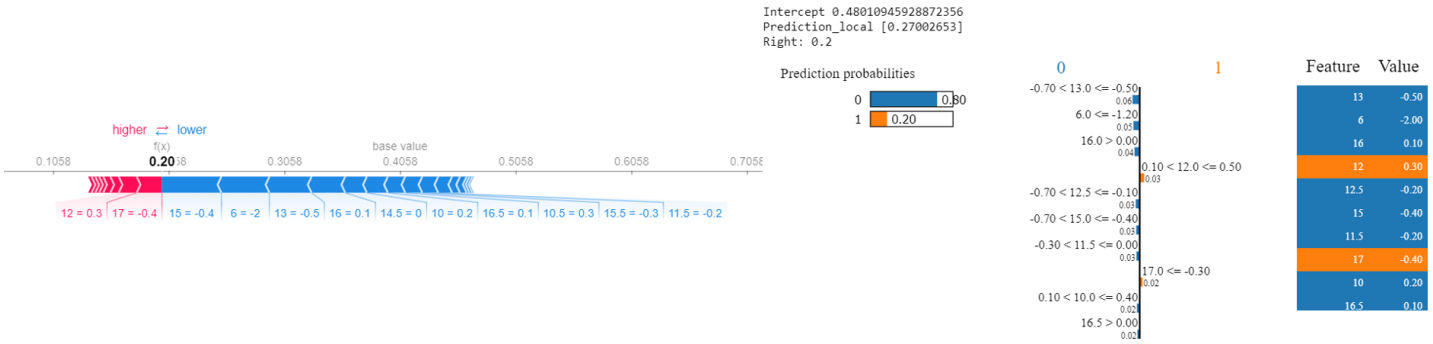
Figure 7.32: Girl number 21



(a) Original function

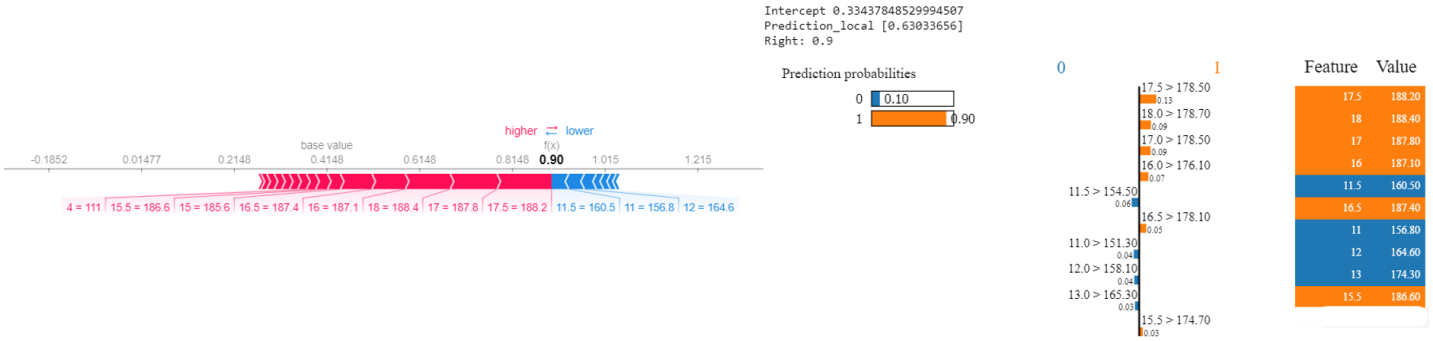


(b) First derivative

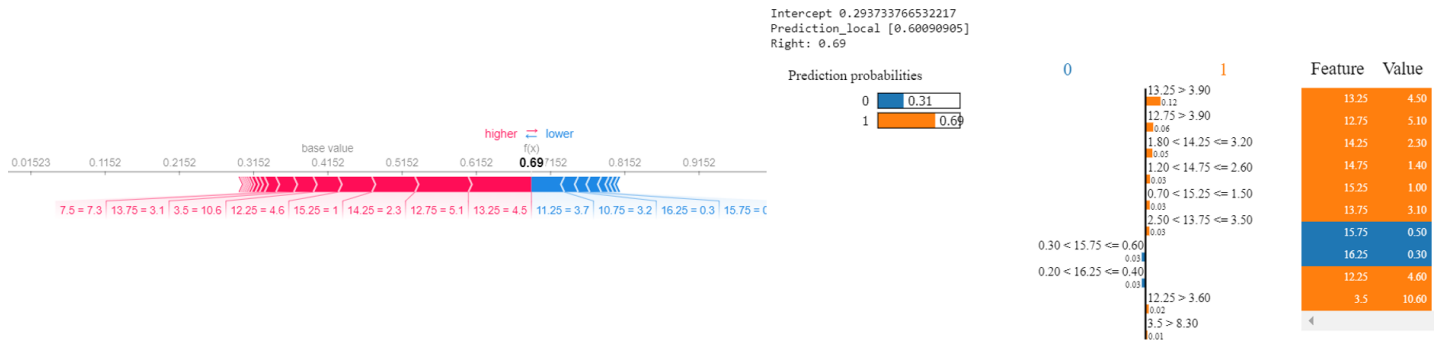


(c) Second derivative

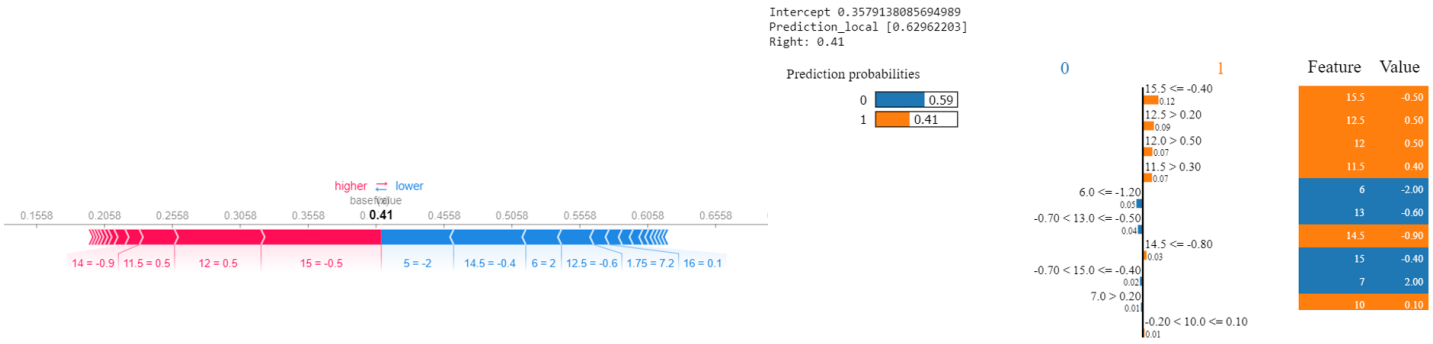
Figure 7.33: Girl number 31



(a) Original function

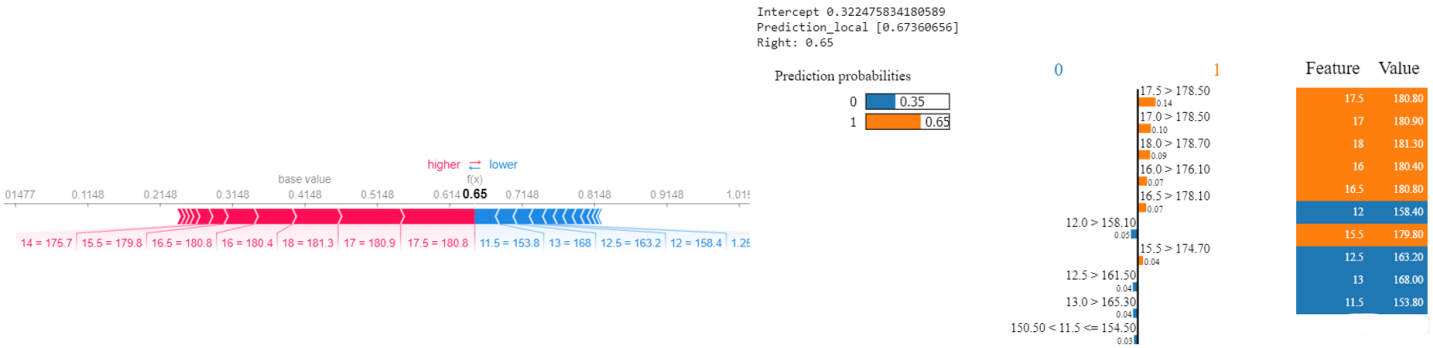


(b) First derivative

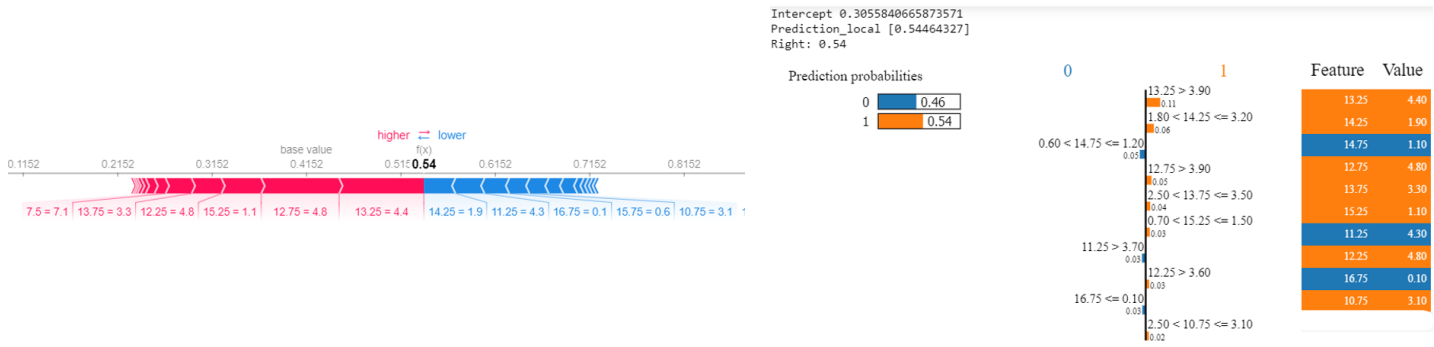


(c) Second derivative

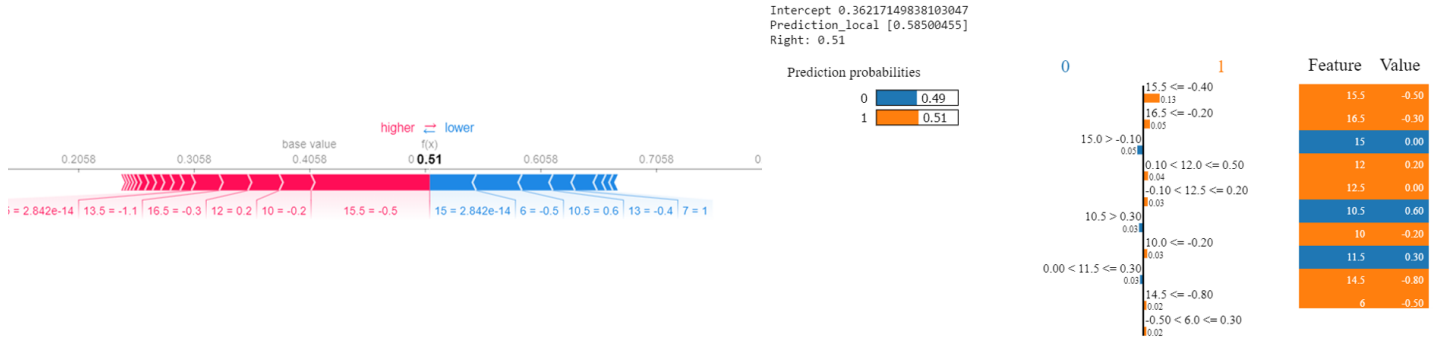
Figure 7.34: Boy number 10



(a) Original function



(b) First derivative



(c) Second derivative

Figure 7.35: Boy number 31

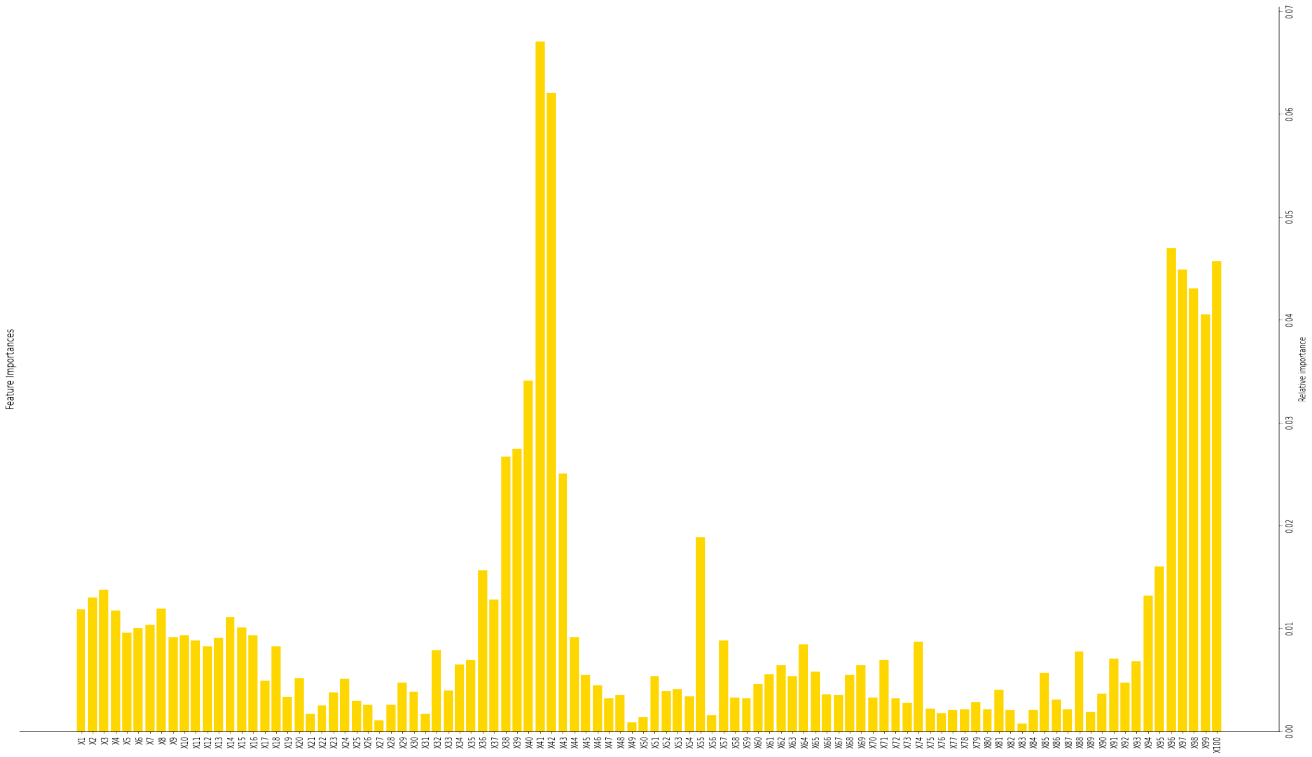
7.3.2. Tecator

At last but not least, it is used the data set of functional data `tecator` in order to construct a random forest and then explain its predictions with LIME and SHAP and also to compare the obtained results with the MDI values. Analog as done with `growth`, three different random forests are going to be constructed: one with the original data, one with the first derivative and a last one with the second derivative.

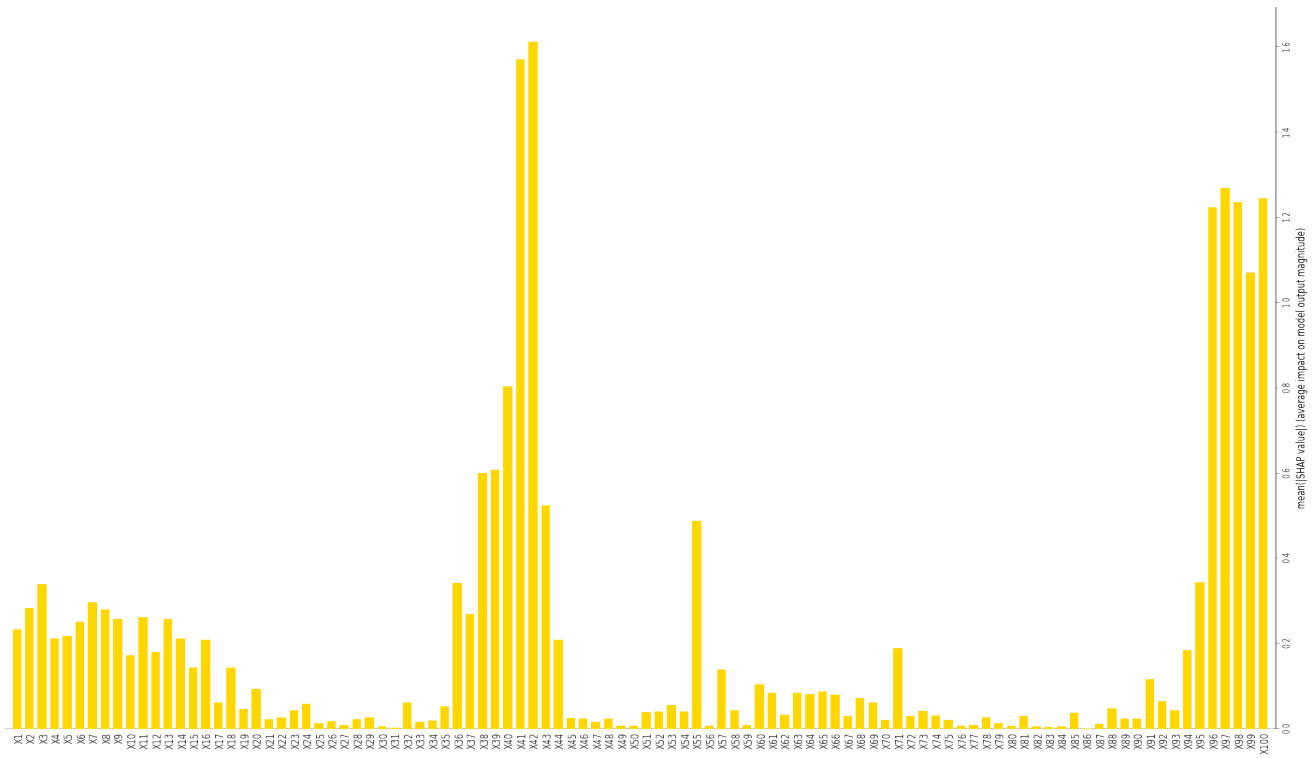
The main difference to the already done experiments is that, this one is not a classification problem, but a regression problem. The predictions of the forest are therefore not a probability between 0 or 1, but a real value. The data set represents the results obtained on 215 meat pieces of an analysis with a food and feed analyzer, called tecator. The response variable is given by the percentages of fat of the meat piece. The explanatory variables correspond to 100 discretization points of the function *Absorbances for the different Wavelenghts* in the interval $[850nm, 1050nm]$. So, when using the original data, such 100 values are the ones used. For the first and second derivatives, the function of the absorbance is differentiated (once and twice, respectively) using the discretization, obtaining 99 and 98 variables, respectively.

Figures 7.39 to 7.43 expose the result of individual instances according to KernelSHAP and to LIME, allowing to establish a comparison of local explanation of both methods and between the cases (a), (b) and (c). First of all, observe that, the individual predictions of the random forests using the first and the second derivative are approximately equal, while the prediction of the random forest using the original function differs in at least a 2% of fat with the lowest prediction of the forests of the derivatives (except for Figure 7.41) and for example in Figure 7.43, even in over a 8%.

Due to the fact that the number of variables used is very high, taking a first look at Figures 7.39 to 7.43 may seem there cannot be stated any similarities or patterns. Nevertheless, taking a look in depth it follows that the by SHAP given higher SHAP values (is absolute terms) features for the first and second derivative mainly agree with the ones that have higher values (in absolute terms) according to LIME, identifying features in the range of Deri43 to Deri45 as important in case of the fist derivative, and in the ranges of 2_Deri30 to 2_Deri41 and 2_Deri95 to 2_Deri98 in case of the second derivative. For the original function, the features differ, i.e. it cannot be identified most of the variables set as important by the one method between the important set by the other one, but there are always variables next to X40 and next to X98 identified as important by both methods, i.e. not exactly the same ones but close features (here it can be spoken about distance between variables because they correspond to the discretization of the absorbance function for concrete wavelengths).

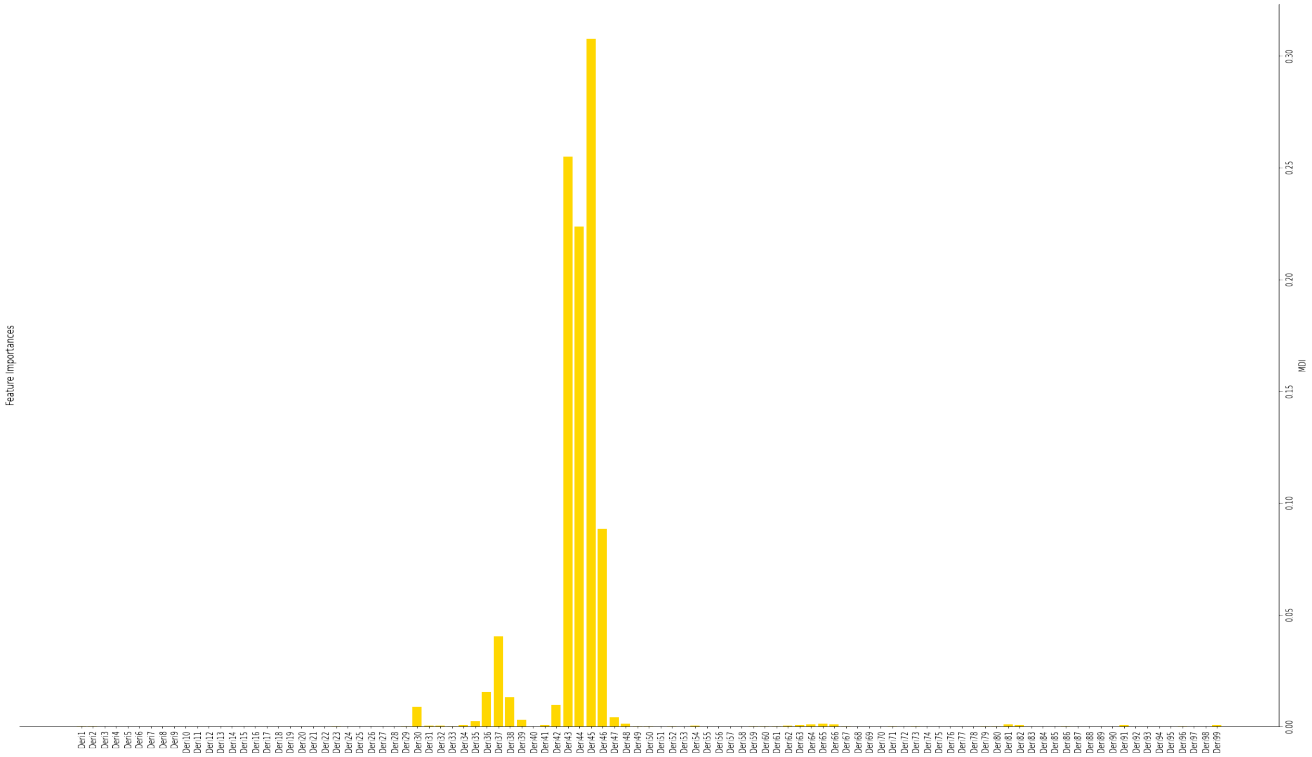


(a) MDI

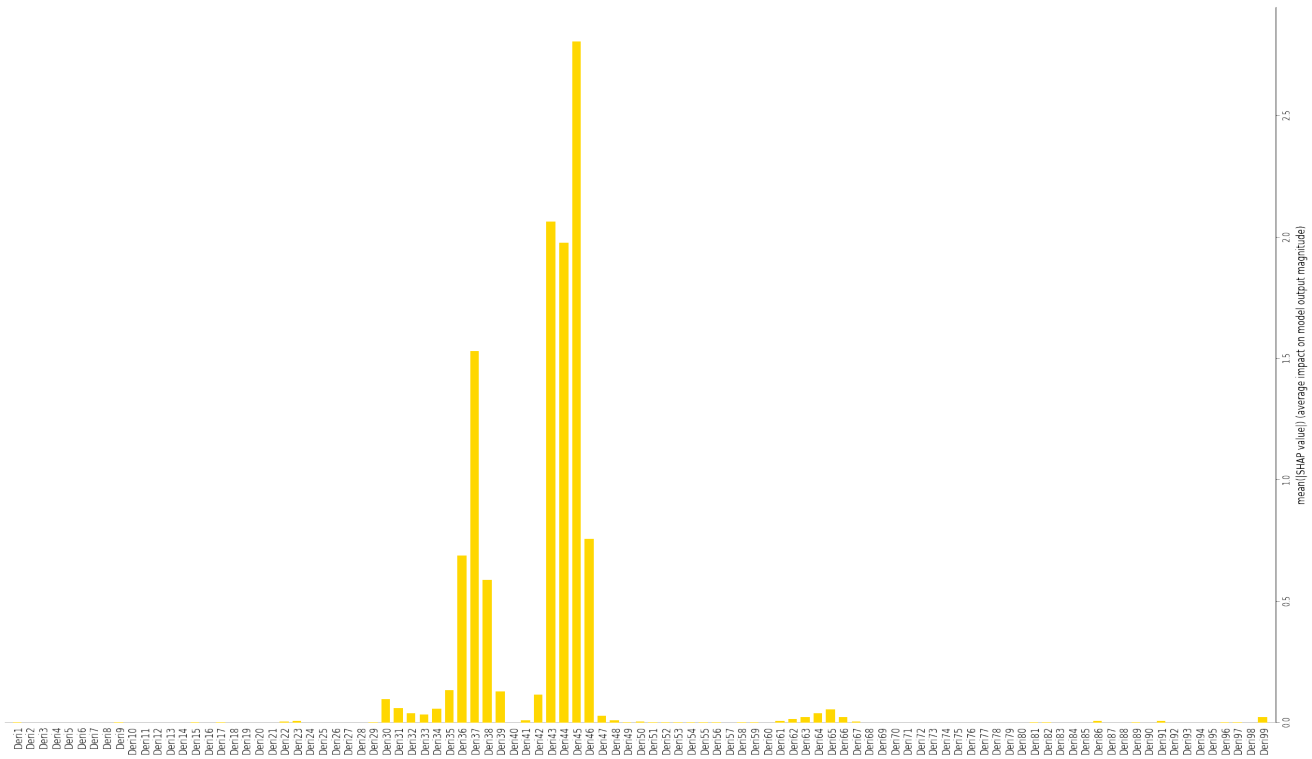


(b) SHAP

Figure 7.36: Global view MDI and SHAP

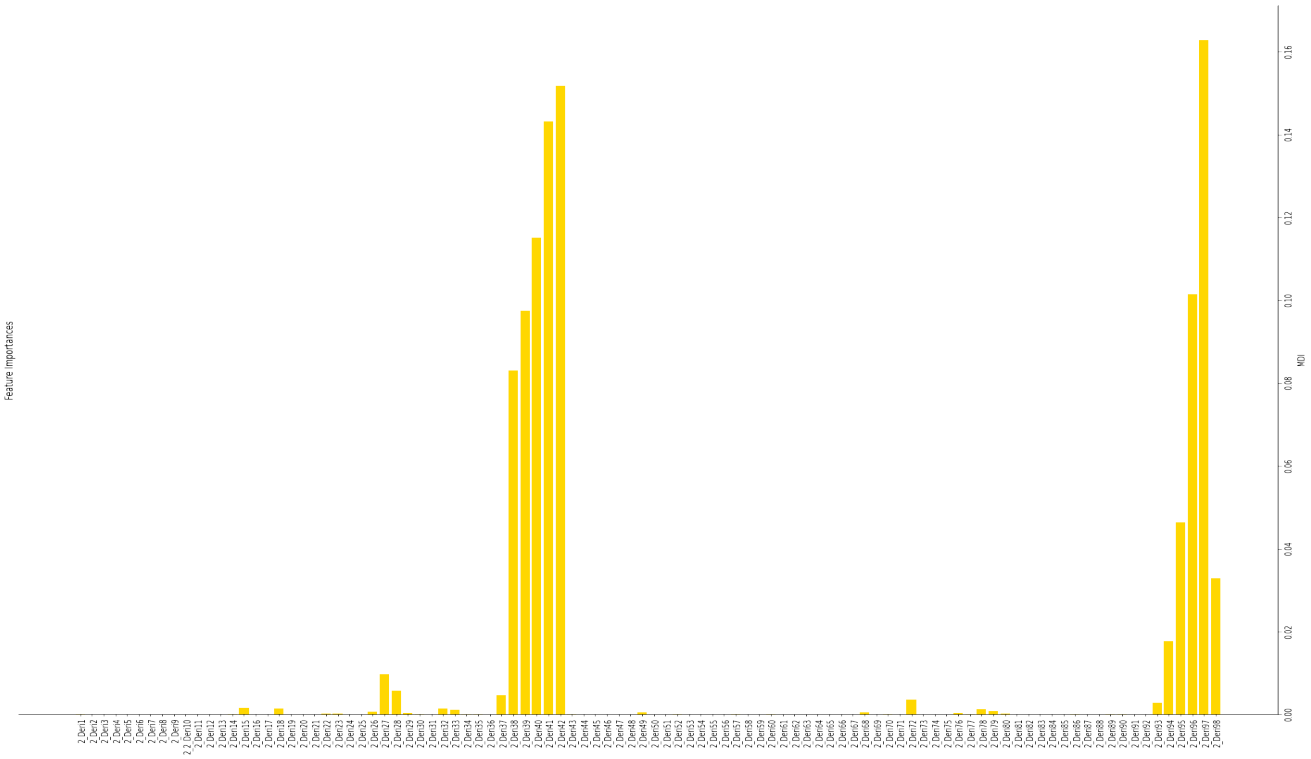


(a) MDI

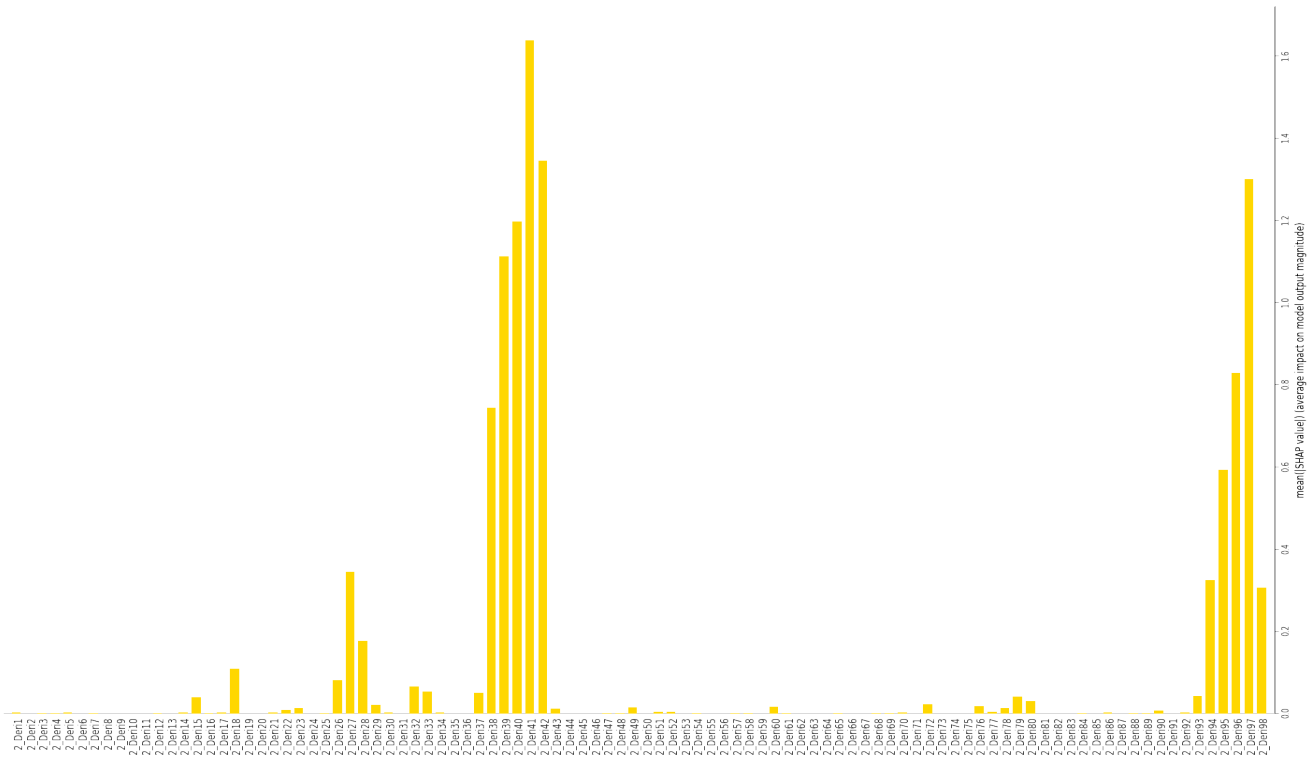


(b) SHAP

Figure 7.37: Global view MDI and SHAP according to first derivative



(a) MDI



(b) SHAP

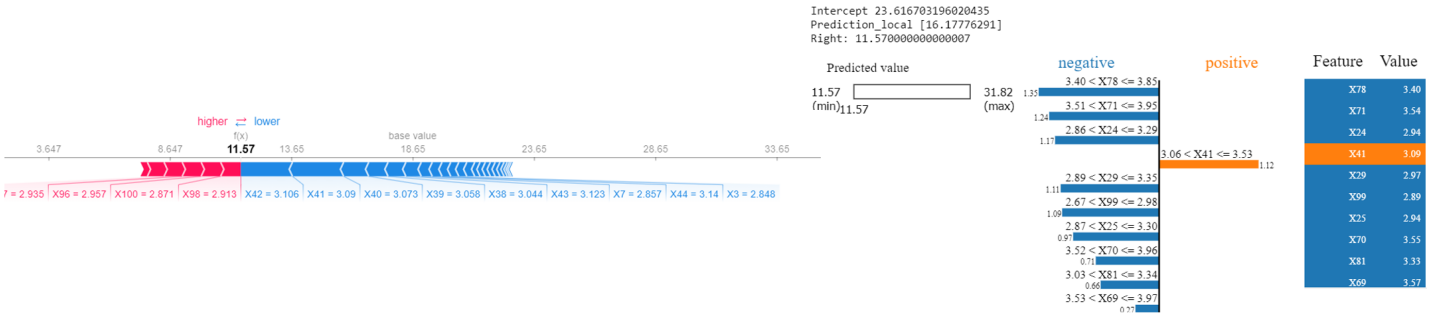
Figure 7.38: Global view MDI and SHAP according to second derivative

Let us next analyze the global view given by SHAP in Figures 7.36 to 7.38 (b). As happened with `growth`, the variables are grouped in valleys and mountains, so that variables that are (un)important correspond to close wavelengths. In particular, in case of using the original function the features that are important, are X36 to X44 and X95 to X100, and in a lesser degree, X1 to X16. In case of the first derivative there are two clearly mountains, corresponding to the variables Deri35 to Deri39 and Deri43 to Deri46. All the other variables have, on average, a SHAP value next to zero. For the second derivative, there are also two mountains corresponding to 2_Deri38 to 2_Deri42 and 2_Deri94 to 2_Deri98.

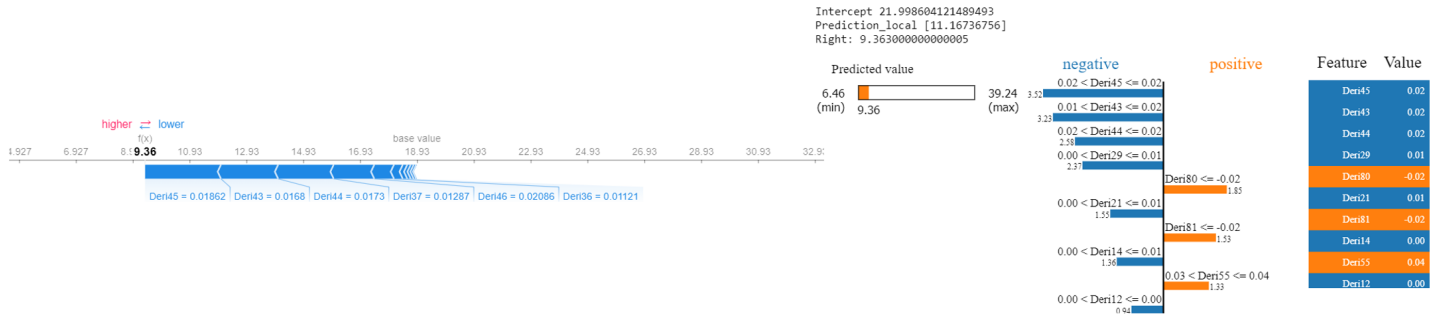
When comparing the SHAP results of Figures 7.36 to 7.38 (b) to the ones of MDI in Figures 7.36 to 7.38 (a) it stands out that, both methods identify the same ranges as (un)important in all three cases.

7.3.3. Conclusions

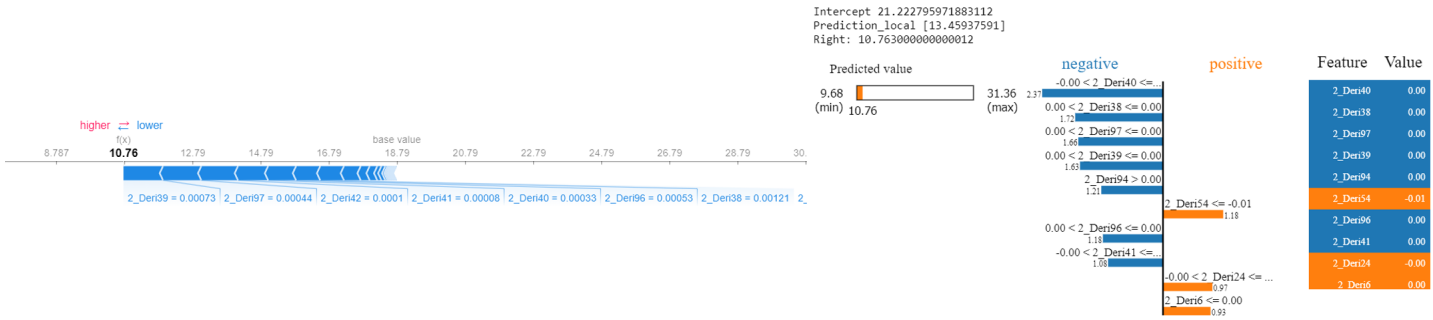
When using functional data, LIME and SHAP give the same importance for local explanations to variables that correspond to close instants of time in case of `growth` and close wavelengths in case of `tecator`. Moreover explanations are mainly similar for both methods, although they can differ. Additionally, the global view of SHAP identifies the valleys and mountains that can be guessed from the few exposed local predictions, sentencing that features in the same range are equally (un)important. Moreover, the results obtained agree with the one submitted by the method based on measuring the impurity MDI.



(a) Original function

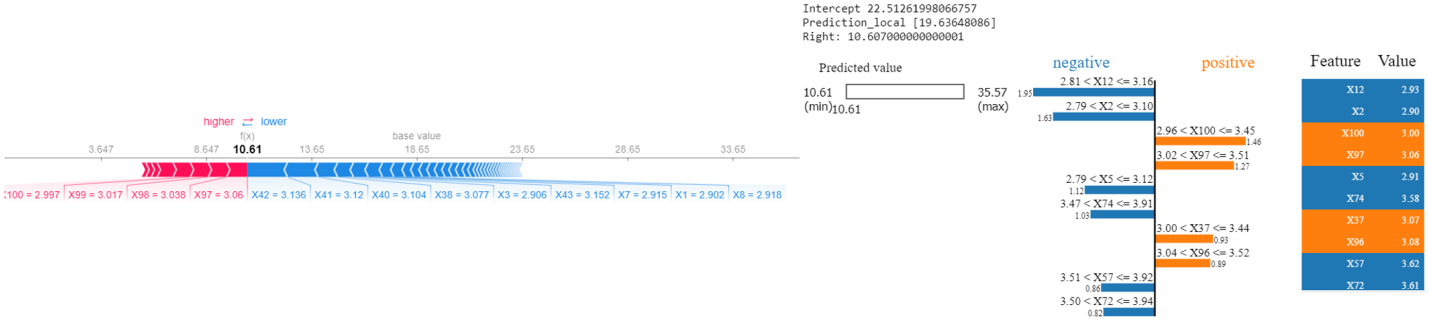


(b) First derivative

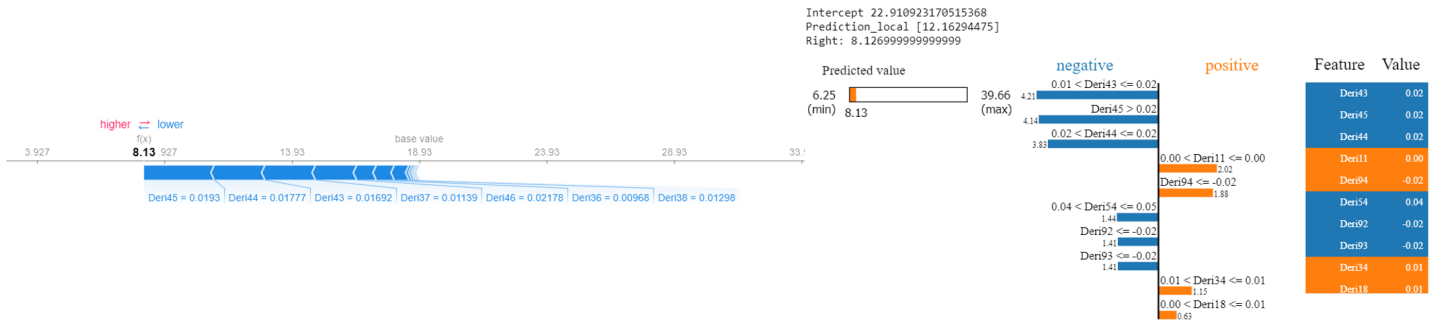


(c) Second derivative

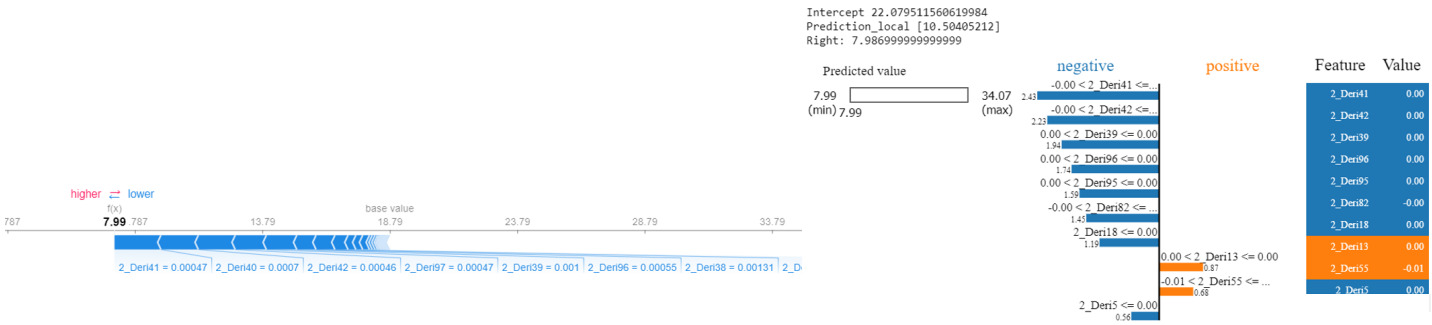
Figure 7.39: Instance 0



(a) Original function

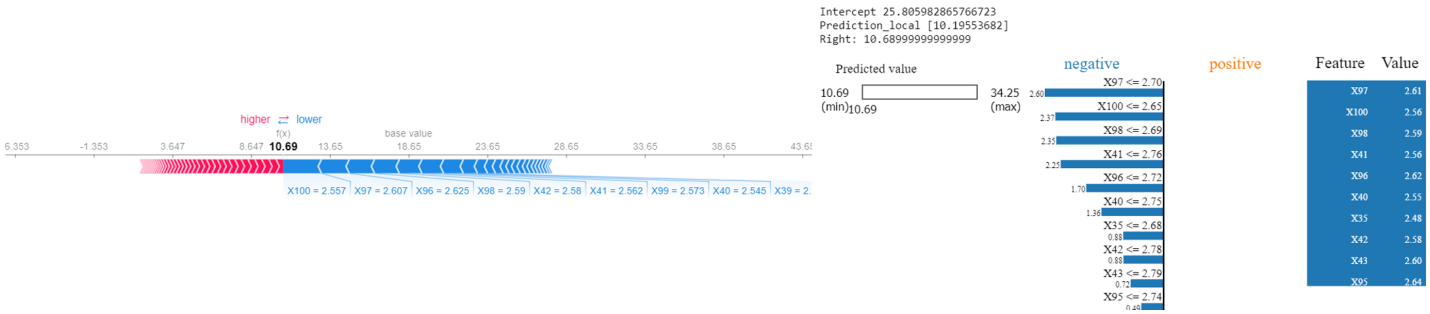


(b) First derivative

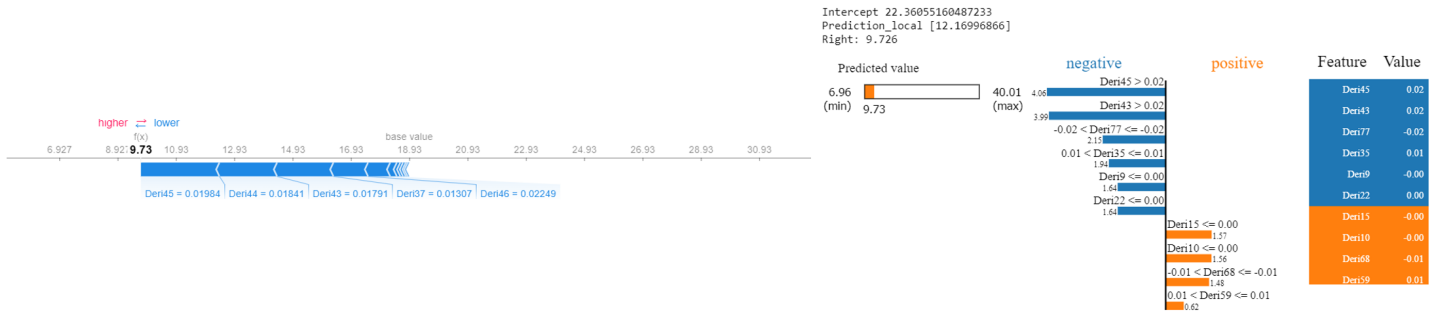


(c) Second derivative

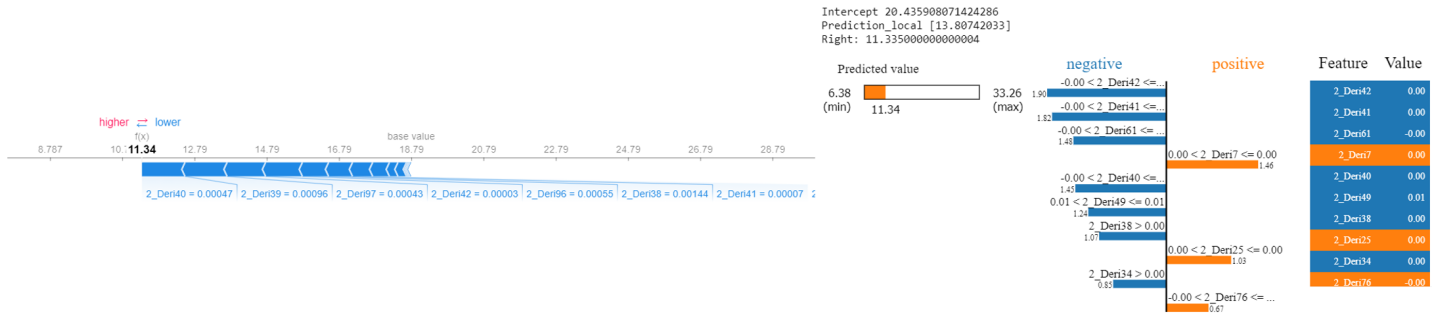
Figure 7.40: Instance 12



(a) Original function

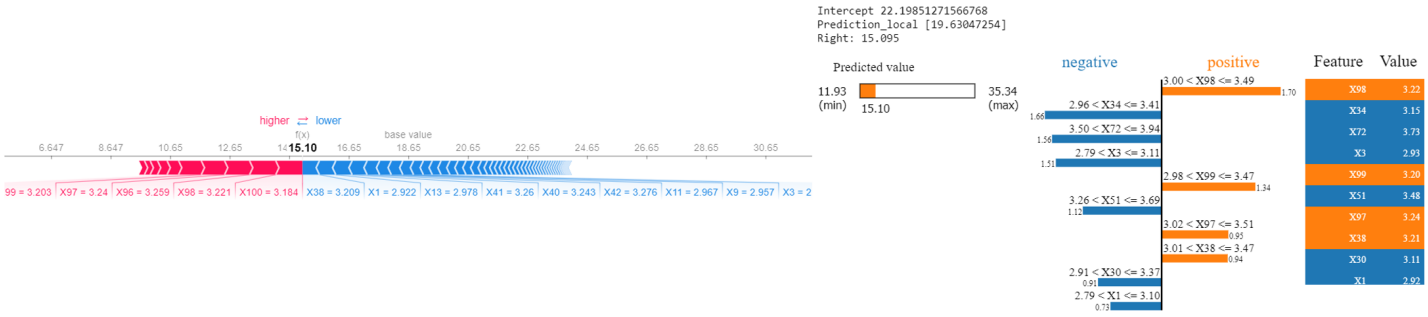


(b) First derivative

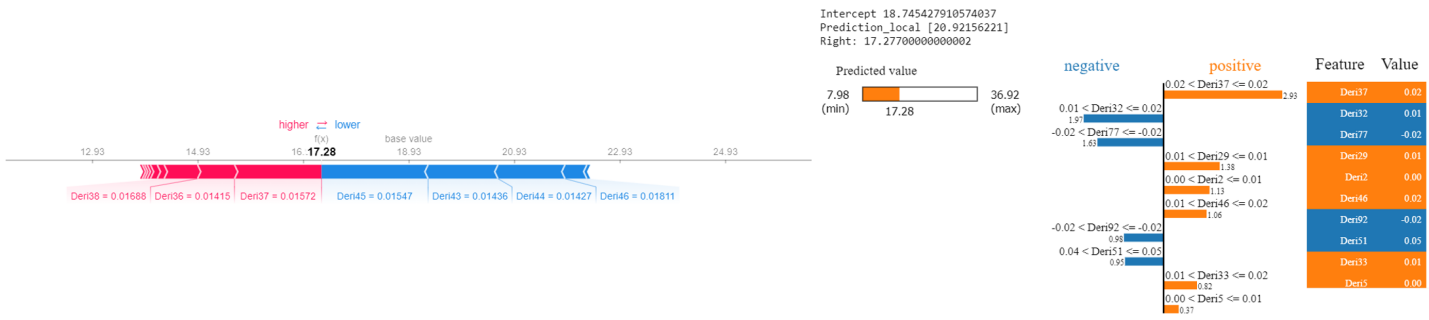


(c) Second derivative

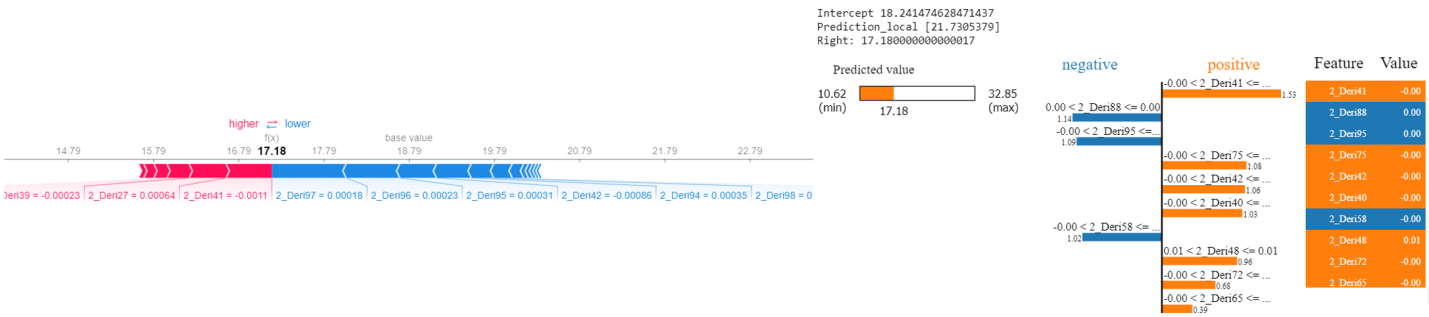
Figure 7.41: Instance 15



(a) Original function

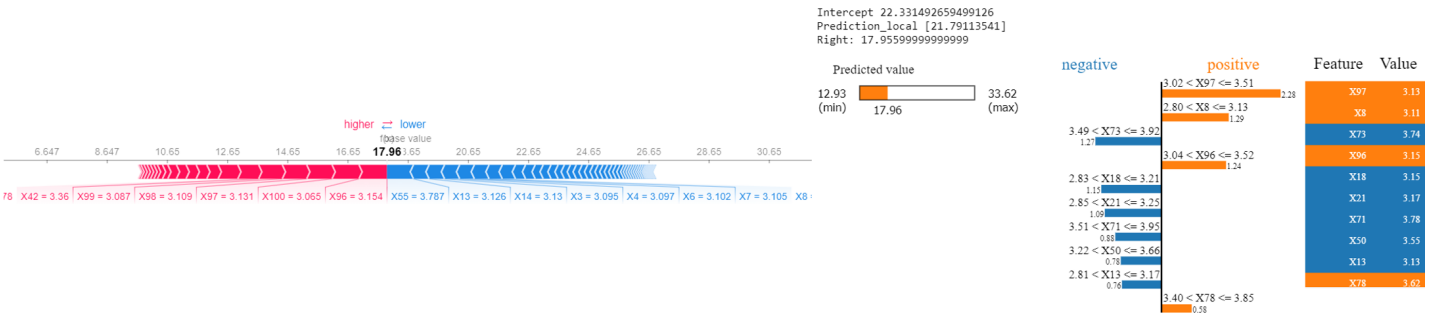


(b) First derivative

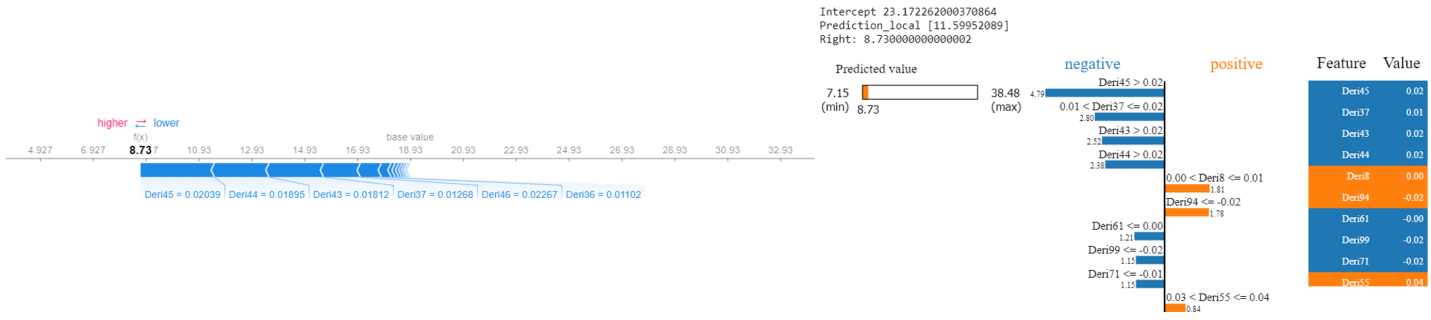


(c) Second derivative

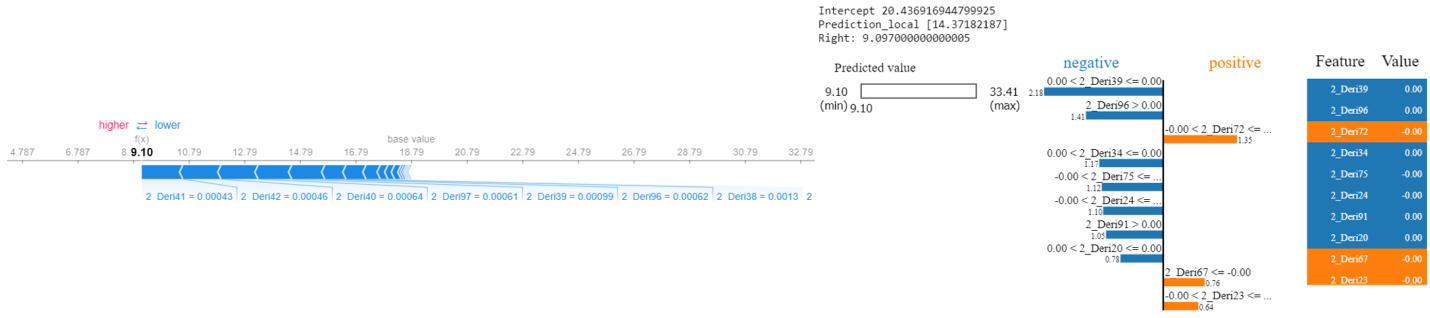
Figure 7.42: Instance 55



(a) Original function



(b) First derivative



(c) Second derivative

Figure 7.43: Instance 60

Chapter 8

Conclusions

Along this text it has been emphasized the need of explanations supporting predictions, because this is the way of ensuring their correctness and fairness, making sure that the model takes ethical decisions, and does not make any assumptions do to ethnic, sex, marital status or any other bias.

Therefore, several explanations methods have been developed, and due to their nature they can be categorized according to three criteria: When is the interpretability obtained? (intrinsic or post-hoc), What is explained? (global or local) and How do they obtain the explanation? (model-specific of model-agnostic). The ones presented in the text, LIME and SHAP fit in the categories of post-hoc and local, but differ in the third category, because LIME is model-agnostic, but SHAP has different variants, among which highlight a model-agnostic one, KernelSHAP and also a model-specific one, TreeSHAP. Moreover, both, LIME and SHAP, fit in the class of additive feature attribution methods, i.e., the explanation given by these methods is the sum of real values that represent the contribution of each feature, rising or lowering the output value from a base value, that in SHAP represents the global average prediction, while in LIME it represents the local average prediction. Furthermore, SHAP is based on the Shapley values, so it is the only additive feature attribution method where the attributions made to each feature fulfill the Shapley values properties of efficiency, symmetry, additivity and null-player. In addition, the KernelSHAP implementation is based on LIME like exposed in Chapter 6.

The different methods have been used to explain predictions of random forests in case of classification as well as of regression in Chapter 7 using the data sets `SouthGermanCredit`, `COMPAS` and the sets of functional data `growth` and `tecator`. The results obtained emphasized that, in most of the cases, the local explanations given by LIME, KernelSHAP and TreeSHAP are very similar, especially when attributing importance to the most important variables, although they are not exactly equal. So, on the one hand, although KernelSHAP is not exact as TreeSHAP for tree models, it is a good approximation, and on the other hand, although KernelSHAP can be seen in

some way as an improvement of LIME, because it connects LIME with the Shapley values, adding its properties, LIME results are not far away. Moreover, SHAP offers the possibility of no only working in a local framework, but also obtaining a global view putting the results of all predictions together.

This way, as well as local as global it is obtained that the random forest used with `SouthGermanCredit` to predict if giving a credit to a person implies or not risk, gives a major effect to the variable `laufkont`, and a lower effect to features like `famges` (marital status and sex) or `gastarb` (foreign worker), so it can be deduced that the local predictions, as well as the whole model, are not discriminatory.
The explanations of the random forests used for `COMPAS` in order to predict if a person will or will not re-offend, using as response variables the real recidivism and afterwards the predictions of the COMPAS algorithm, show that in both cases the most important variable is `priors_count`, but also in both cases, there is attributed some effect to variables like sex age or race, especially when using the predictions of the COMPAS algorithm. This example stands out how important it is, to know the explanation behind the prediction, because they are not necessary fair, and that the choose of features to use with a model is also important, because instead of race, perhaps it should be used variables like incomes, job, etc.
At last but not least, the experiments with the data sets `growth` and `tecator` emphasize that according to global view of SHAP the (un)important variables are grouped together, i.e. not concrete "instants" of time or space are important, but intervals, so that, variables that are (un)important are close to each other, forming this way mountains (the important ones) and valleys (the unimportant ones). Moreover, these results are very similar in case of the random forest for the given by MDI. Furthermore, locally the results of SHAP and LIME do also mostly agree.

Glossary

Symbols	Definition
$\mathcal{D}$	learning sample
M	number of observations in $\mathcal{D}$
N	set of features
$\|N\|$	number of features in N
N'	set of features in the simplified space
$\|N'\|$	number of features in N'
$\mathcal{X}$	measurement space
$\mathbf{X}^j$	$j-th$ sample of N variables of $\mathcal{D}$
X_i^j	$i-th$ features of $\mathbf{X}^j$
Y^j	$j-th$ observation of $\mathcal{D}$
$\mathcal{C}$	set of classes
R	number of possible classes
C_r	class r of C
t_i	node i
H	number of nodes a nodes splits into
$p(r\|t)$	probability of an observation to be of class C_r subject to having reached node t
u	split
$\delta i(v,t)$	decrease in impurity by splitting node t by split v
f	prediction model
g	explanation model
x	original input to be explained
x'	simplified input in binary space
h_x	mapping function from simplified space to original space
z	perturbation of x
z'	perturbation of x'
$\mathcal{Z}$	set of perturbed instances
ϕ	coefficients of g
	in SHAP they are the SHAP values
G	class of interpretable models
$\Omega(g)$	complexity of g
$\pi_x(z)$	local kernel, proximity measure

	between x and z
D	distance measure
σ	kernel-width
$\mathcal{L}$	locality-aware loss
q_k	quartiles
S	set of non-zero entries in z' (coalition)
$\|S\|$	size of S
$v(S)$	set function, pay-off
$f_x(S)$	$= f(h_x(z'))$
z_S	vector z where the entries equal to 1 are the ones of S
$z_{\bar{S}}$	vector z where the entries equal to 1 are the ones not in S
T	number of trees
F	maximum depth
L	number terminal nodes
$\Phi_{i,j}$	interaction value of features i and j
$\nabla_{i,j}$	assistant function in $\Phi_{i,j}$

Bibliography

[1] *Cooperative Game Theory Tools in Coalitional Control Networks*. Springer International Publishing, 2019.

[2] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values, 2020. https://arxiv.org/pdf/1903.10464.pdf.

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

[4] Guilherme Alves, Vaishnavi Bhargava, Miguel Couceiro, and Amedeo Napoli. Making ML models fairer through explanations: the case of LimeOut. In *9th International Conference on Analysis of Images, Social Networks, and Texts 2020 (AIST 2020)*, Lecture Notes in Computer Science, Analysis of Images, Social Networks, and Texts 2020, Moscow, Russia, October 2020.

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[6] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.

[7] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification And Regression Trees*. CHAPMAN HALL/CRC, 1984.

[8] Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey, 2019. https://arxiv.org/pdf/1911.12116.pdf.

[9] Emilio Carrizosa and Dolores Romero Morales. Supervised classification and mathematical optimization. *Computers Operations Research,40(1),150-165*, 2013.

[10] Carlos María Romeo Casabona. Riesgo, procedimientos actuariales basados en inteligencia artificial y medidas de seguridad*. *Revista de derecho, empresa y sociedad*, 2018.

[11] Hugh Chen, Joseph D. Janizek, Scott Lundberg, and Su-In Lee. True to the model or true to the data?, 2020. https://arxiv.org/pdf/2006.16234.pdf.

[12] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning, 2019. https://arxiv.org/pdf/1808.00033.pdf.

[13] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1):72–99, 2006.

[14] Damien Garreau and Ulrike von Luxburg. Explaining the explainer: A first theoretical analysis of lime, 2020. https://arxiv.org/pdf/2001.03447.pdf.

[15] Damien Garreau and Ulrike von Luxburg. Looking deeper into tabular lime, 2020. https://arxiv.org/pdf/2008.11092.pdf.

[16] Riccardo Guidotti. Evaluating local explanation methods on ground truth. *Artificial Intelligence*, 291:103428, 2021.

[17] Alan Julian Izenman. *Modern Multivariate Statistical Techniques. Regression,Classification, and Manifold Learning*. Springer, 2013.

[18] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem, 2019. https://arxiv.org/pdf/1910.13413.pdf.

[19] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

[20] Scott Lundberg, Gabriel Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. 02 2018. https://arxiv.org/pdf/1802.03888.pdf.

[21] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. 12 2017. https://arxiv.org/pdf/1705.07874.pdf.

[22] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable ai for trees: From local explanations to global understanding, 2019. https://arxiv.org/pdf/1905.04610.pdf.

[23] Dina Mardaoui and Damien Garreau. An analysis of lime for text data, 2020. https://arxiv.org/pdf/2010.12487.pdf.

[24] Andreas Messalas, Yiannis Kanellopoulos, and Christos Makris. Model-agnostic interpretability with shapley values. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–7, 2019.

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

[26] Christoph Molnar. *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*.

[27] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning – a brief history, state-of-the-art and challenges. 10 2020. https://arxiv.org/pdf/2010.09337.pdf.

[28] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[29] M.R. Ortiz-Posadas. *Pattern Recognition Techniques Applied to Biomedical Problems*. Springer.

[30] Erika Puiutta and Eric M. S. P. Veith. Explainable reinforcement learning: A survey. In Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 77–95, Cham, 2020. Springer International Publishing.

[31] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. pages 97–101, 02 2016.

[32] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges, 2021. https://arxiv.org/pdf/2103.11251.pdf.

[33] Cynthia Rudin and Joanna Radin. Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition. *Harvard Data Science Review*, 1(2), 11 2019. https://hdsr.mitpress.mit.edu/pub/f9kuryi8.

[34] Cynthia Rudin, Caroline Wang, and Beau Coker. The age of secrecy and unfairness in recidivism prediction, 2019. https://arxiv.org/pdf/1811.00731.pdf.

[35] Ludwig Schallner, Johannes Rabold, Oliver Scholz, and Ute Schmid. Effect of superpixel aggregation on explanations in lime – a case study with biological data, 2019. https://arxiv.org/pdf/1910.07856.pdf.

[36] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution.". *BMC bioinformatics*, 8(1):25, 2007.